No. 1
January 1985
£1

A Database Publication

Computing with the AMSTRAD

The independent magazine for CPC 464 users

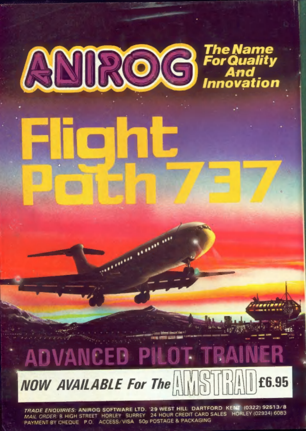
Games galore!

Save Smiley from the Grumpies
Help trap the maze monsters
Crack the secret code
...and more!

Have fun with the Amstrad's superb colour and sound

14 software releases reviewed

Amstrad teach-in Baffled by Basic...mystified by machine code? We show how easy they really are!


ANIROG
The Name For Quality And Innovation
Flight Path 737
ADVANCED PILOT TRAINER
NOW AVAILABLE For The AMSTRAD £6.95
TRADE ENQUIRIES: ANIROG SOFTWARE LTD. 29 WEST HILL DARTFORD KENT (0322) 92513/8
MAIL ORDER: 8 HIGH STREET HORLEY SURREY 24 HOUR CREDIT CARD SALES HORLEY (02934) 6083
PAYMENT BY CHEQUE P.O. ACCESS/VISA 50p POSTAGE & PACKAGING

SUPERB SOFTWARE FOR THE AMSTRAD
SUPER-FAST LOADING TIME commodore 64
electron
B.B.C. MICRO
GHOULS
Run through the creepy mansion dodging ghostly ghouls and bouncing spiders. Leap over poison-smeared spikes, scamper along moving platforms and contracting floorboards, and use powerful springs to propel you onto overhanging ledges. Four screens.
Amstrad and Commodore versions £6.95
BBC and Electron versions £7.95
BBC and Commodore Disk price £9.95
Amstrad version
WATCH OUT FOR OUR NEW PACKAGING AND CATALOGUE
MICRO POWER
MICRO POWER LTD., NORTHWOOD HOUSE, NORTH STREET, LEEDS LS7 2AA TEL: (0532) 458800
SELECTIVE BRANCHES OF BOOTS, CO-OP, MENZIES, W. H. SMITH, WOOLWORTHS AND ALL GOOD DEALERS
AUTHORS! WE PAY 20% ROYALTIES!

Vol. 1 No. 1 January 1985

Managing Editor: **Derek Meakin**
Features Editor: **Peter Bibby**
The A Team: **Mike Bibby**
Alan McLachlan
Kevin Edwards
Production Editor: **Peter Glover**
Layout Design: **Heather Sheldrick**
News editor: **Mike Cowley**
Advertisement Manager: **John Riding**
Advertising Sales: **Margaret Clarke**
Editor in Chief: **Peter Brameld**

Editorial: 061-456 8835
Administration: 061-456 8383
Advertising: 061-456 8500
Subscriptions: 061-480 0171
Telex: 667664 SHARET G
Prestel Mailbox: 614568383

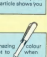
Published by:
Database Publications Ltd,
Europa House, 68 Chester Road,
Hazel Grove, Stockport SK7 5NY.

Subscription rates for 12 issues, post free:
£12 – UK
£15 – Eire (Sterling only)
£20 – Rest of world (surface)
£40 – Rest of world (airmail)

ABC Member of Audit Bureau of Circulations

"Computing with the Amstrad" welcomes program listings and articles for publication. Material should be typed or computer-printed, and preferably double-spaced. Program listings should be accompanied by cassette tape or disc. Please enclose a stamped, self-addressed envelope, otherwise the return of material cannot be guaranteed. Contributions accepted for publication by Database Publications Ltd will be on an all-rights basis.

News trade distribution:
Europress Sales and Distribution Limited, 11 Brighton Road, Crawley, West Sussex RH10 6AF. Tel: 0293 27053.



Contents

Introducing ourselves...

WELCOME to the first issue of Computing with the Amstrad – the new monthly magazine written by users of the CPC464 for users of the CPC464.

We're big, bright and completely independent, so what you read here is totally impartial. For instance, our program reviews tell it like it is – not how the software houses would like it to be.

Actually, we're quite impressed with the standard of software available for the CPC464 as you'll see from our review pages.

Our one quibble is that, good though the programs are, too many show signs of having been quickly translated from other computers. All too often they fail to take full advantage of the Amstrad's unique features and potential.

But you'll find much more than reviews inside this issue. As you'll see from this page, the magazine's packed with original programs and informative articles.

We know that many of you are newcomers to micro-computing, so we've got plenty to interest the novice – including easy-to-follow introductory articles on the Amstrad's Basic, Sounds and Graphics.

And we've not forgotten those of you with more experience. You'll find plenty to interest you, including an introduction to machine code and a survey of the Amstrad's hardware, as well as lots of useful ideas and routines you can incorporate into your own programs.

One thing we're sure of is that you've got your own ideas of what the magazine should cover. We're always open to suggestions – so let's have yours.

And we're on the lookout for writers, too. If you've got a program you think would interest us let us know. You won't make a fortune, but you'll have a lot of fun.

And thanks for all the letters – keep them coming.

See you next month.

The A Team

Amstrad Adventures from Interceptor Software

R.R.P. £6.00 EACH

Interceptor Software are proud to present their first 4 releases on the Amstrad. These 'user friendly' graphical adventures not only utilise the memory to a full but display the best graphics yet seen on this new computer. A sensible price of £6.00 each makes them the best value software on the Amstrad.


Forest at Worlds End

Forest at Worlds End is a mythical adventure where you have to rescue Princess Mara who has been captured by the evil wizard, Zarn. Many foes await you in the forest!!


Heroes of Karn

Heroes of Karn the hit adventure on the CBM 64 has now been converted to the Amstrad. This classical story will keep you occupied for many moons and when you finally solve the adventure and find the Heroes you will find a vacuum in your life only to be filled with the follow on, Empire of Karn which will be released in 1985.


Jewels of Babylon

Jewels of Babylon is set with a pirate theme, where you, the sole survivor of a pirate raid have the task of recovering the Jewels for Queen Victoria who has promised them as a wedding gift to an Indian Princess


Message From Andromeda

Message From Andromeda is a space adventure where you are a captain of a patrol cruiser in the galaxy. The task set for you is laid fraught with danger? Be prepared for the unexpected.

Available from all leading Software Retailers or direct from

INTERCEPTOR SOFTWARE Lindon House, The Green, Tadley, Hants, England.

Tel:(07356) 71145/3711 Telex: 849101 INMICS G

Stand back, sliced bread

IS Amstrad the greatest thing since sliced bread? Inside the company they've no doubts.

Company chairman Alan Sugar says: "In all the years I have been part of this industry I have never experienced such a demand for any one product.

"The enquiries from home and overseas have been quite overwhelming and supply is nowhere near demand".

Ray Cope, head of the computer division at distributors Europa Electronics, says: "I have never in all my experience in distribution and sales seen a product take off like this".

Alan Sugar: "Never experienced such a demand".

Sophisticated new projects on way

BEHIND the CPC464 stands a highly profitable international organisation. Group turnover of Amstrad Consumer Electronics in 1983/84 was £84.9 million – up more than £33 million from the previous year.

Group pre-tax profit was £9.1 million compared with £8 million, and exports more than doubled from £5.3 million to £10.8 million.

These figures cover sales of the company's racked audio units, colour TVs and video recorders and were compiled before the CPC464 appeared in the shops.

Exciting

Projected computer sales are 200,000 in 1984 and 600,000 worldwide in 1985 – together with peripheral hardware to complement the machine.

In his annual report, Amstrad chairman Alan Sugar promised: "There are exciting plans under development in this sector for new items to be launched in the third quarter of 1985.

"These new projects will incorporate more sophisticated electronic technology and keep Amstrad ahead of the market".

Campaign postponed

THE CPC464 has made a huge impact on the computer marketplace with very little advertising backup.

Demand for the machines has been so great that a planned £2 million nationwide TV campaign has been postponed until early 1985.

Instead, Amstrad has been restricting its advertising to national daily papers and specialist magazines.

The reason, says a spokesman, is "the Amstrad reputation for value for money products is second to none.

"Many faithful customers owning other Amstrad products are now buying the computer.

"Since we started delivery of the first computers in June, the product has made a wonderful reputation with both trade and user.

"To have continued with our TV advertising in the autumn would have complicated the supply position.

"But by a big blast starting in the New Year we can be assured of a good spring sales level".

Did you know?

AMSTRAD gets its name from the abbreviation of Alan M. Sugar Trading, which started manufacturing electronic equipment in 1968. Alan Sugar is the present chairman and chief shareholder of Amstrad.

Wally moves in

WALLY Week, star of Micro-Gen's Automania and Pyjamarama, has set his sights on becoming a favourite of Amstrad users. Pyjamarama is now available for the CPC464 at £8.95.

Fairytale adventure

READ the story, play the game – that's the theme of a novel Amstrad package from Database Publications.

The Magic Sword is an adventure game on cassette, specially written for young children. But that's only half the story.

With it comes a colourful 48 page book which tells all the events leading up to the start of the adventure.

All the elements of a traditional fairy story are there – a handsome prince, a beautiful princess, a white castle with secret passages and mysterious dungeons, dense forests, deep woods – and a crooked house where lives a wicked witch.

All these are in the game as well, plus colourful animated graphics and lots of thrilling sound effects which encourage the child to travel through the countryside and explore the castle to find the princess and release her from the clutches of the witch.

The program is the brain child of housewife Kristin Hollis and her son Martin, 13.

Kristin wrote the book with help from Martin, who drew the pictures. Martin devised the game program based on his mother's design and graphics.

Goodies galore lined up

FOR a completely new machine, the CPC464 is surprisingly well provided with peripherals and software.

☐ Amstrad's own 80 column dot matrix printer, uses a standard parallel centronics interface for text processing.

The DMP-1 operates at a print speed of 50 cps and includes instruction extensions that provide dot-addressable graphics and full screen dump capability.

The self-aligning tractor feed is adjustable from 4.5in to a maximum paper width of 10in.

☐ A printer lead, price £9.95, is available for use on non-Amstrad printers.

☐ Selling at £29.95, the Amstrad power modulator allows the CPC keyboard to be used with a domestic colour TV set.

☐ The Amstrad joystick's design allows two to be operated from one port. Cost is £14.95 a unit.

☐ The Amstrad disc drive costs £199.99 and comes complete with interface, PSU, CP/M and Dr Logo operating systems.

☐ Firmware and Basic specification manuals retail at £19.95 each.

Amsoft, the specialist division set up to market the software support has more than 80 arcade style games available now or due for release shortly. Most cost £8.95.

About a dozen educational and tutorial programs are available, ranging in price from £8.95 to £35.

Amstrad drive for schools market

AMSTRAD is going all out to capture a major share of the education market from Acorn and RML.

And if all goes to plan over the next few months the CPC 464 could become the schools' computer of the future.

Amstrad admits it is currently short of software ammunition in the educational sector – a situation it aims to quickly rectify – but insists its prices and product are potent weapons.

First move in the campaign was the recent appointment of Northern Computers as exclusive educational distributor for the CPC464.

Northern's education and training division director Gareth Littler has drawn up his battle plan.

Primary target is the ageing RML 480Z whose capabilities are "virtually matched by the CPC at half the price" according to Littler.

Convert

Northern is asking all publishers of RML educational software to convert their programs for the Amstrad. "Not a very difficult task given the similar capabilities of the two machines", says Littler.

His next task is to get 700 BBC software publishers to do likewise. "They will have to re-write the programs, but will have twice the memory to play with", says Littler.

In a third line of attack Northern has already directly approached all Britain's 35,000 schools, colleges and universities – giving them information about the product and Amstrad's plans for its future.

Northern Computers' Gareth Littler – getting a foot in the classroom door

Here the main selling point has been price. Schools can buy a green screen CPC464 for £189 or a colour version for £279.

"We can supply a lab with 18 colour Amstrads for what it costs to buy 10 BBC Micros with colour", says Littler.

"We have already placed large numbers of machines in educational establishments, with a particularly big response from polytechnics.

"Another big selling point is our free one year service and maintenance contract".

Publicity in the educational press has been particularly helpful. "The machine has had a wonderful reception from educationalists", says Littler.

Another big boost will come very shortly when Amstrad brings out its interface for Econet, enabling the CPC464 to join BBC Micro networks in schools.

Amstrad is also tackling the language and CP/M compatibility of its machine.

First drives to be made available are the 3in version with CP/M and Logo.

But soon the CPC464 will have Micropulse 5¼in disc drives, also with CP/M and Logo.

Rush to board the bandwagon

THE runaway success of the CPC464 left many computer dealers biting their nails as the Christmas sales period loomed.

The reason – their applications to climb onto the fast rolling Amstrad bandwagon were piling up at the offices of Europa Electronics, sole distributors to independent retailers.

With the seasonal spending spree about to begin, 450 dealers had received their credit clearance and been registered by Europa.

But there was a backlog of about 100 applications from retailers desperate to stock the fast-selling machines.

Ray Cope, head of the Europa computer division explained: "A major cause of the delay in processing applications is that so many come from dealers we haven't worked with before and they nominate our competitors as credit references.

"I'm sure our competitors would not deliberately sit on these applications – it's more likely a case of us overloading them with paperwork".

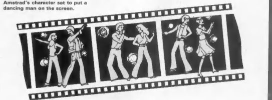
Europa had planned to release 25,000 machines to independent dealers but Amstrad was hoping to boost this total to 40,000.

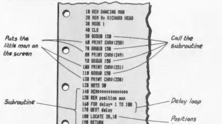
Added to the micros supplied to major chains like W.H. Smith, Boots, Rumbelows and Comet, it adds up to a projected 1984 sales figure of 200,000 for the CPC464.

ANALYSIS is the first of a regular series where short, simple programs come under scrutiny. This month's program uses the Amstrad's character set to put a dancing man on the screen.

Amstrad Analysis

By Trevor Roberts





```
10 REM DANCING MAN
20 REM By RICHARD HEAD
30 MODE 1
40 CLS
50 GOSUB 150
60 PRINT CHR$(250)
70 GOSUB 150
80 PRINT CHR$(249)
90 GOSUB 150
100 PRINT CHR$(251)
110 GOSUB 150
120 PRINT CHR$(250)
130 GOTO 50
140 REM***************
150 REM position man
160 FOR delay= 1 TO 100
170 NEXT delay
180 LOCATE 20,10
190 RETURN
200 REM***************
```


10,20	Two REM statements saying what the program is called and who wrote it.
30	Selects Mode 1.
40	Clears the screen.
50,70,90,110	Call the subroutine which starts at line 150 and ends at 190.
60,80,100,120	These lines do the work of printing the little man on the screen. The differing figures in the brackets of the CHR$() produce a different little man.
130	The GOTO sends the program back to line 50, causing the program to repeat endlessly.
140,200	The REMs and asterisks are just used to show where the subroutine is hidden.
150-190	Form the subroutine.
150	Makes a note of what the subroutine does.
160-170	The FOR . . . NEXT loop just slows things down a little so we have time to see the man.
180	The LOCATE makes sure that the man is printed at the same position each time the subroutine is called. As the different characters overprint each other, so the man appears to dance.
190	The RETURN ends the subroutine, sending the Amstrad back to the line following the line that called the subroutine.

Programming is easier than you think, and doubly so if you follow MIKE BIBBY's crystal clear guide through the micro jungle

I DON'T know who you are. You might be a wife whose CPC464-owning husband is away at work, or a father who is trying to come to terms with his daughter's Christmas present.

Alternatively you might be a teacher who has just been "computerised". Whoever you are, the the fact that you are reading this article tells me your guilty secret – you want to be able to program the Amstrad CPC464.

But how to begin? You must have noticed that some people take to computing like ducks to water, in an output port to an interface as they would say. Words like byte, strings and user-defined characters flow freely from their lips. They pass parameters and handle interrupts with ease, then get their hands on and peek and poke in a way that beggars belief!

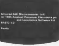
You, I take it, are not like that. You are not a computer "natural". But you would dearly like to be. Well fear not, this series of articles is for you, and it was written by one of your kind.

I, too, have sat at the keyboard, staring at the cursor, without having any idea of what to do next (or even knowing it was called a cursor). I also know what it's like to have someone explain to me in the "simplest possible terms" and still find it way above my head. Yet I now program reasonably well . . . and so can you. Read on!

LET'S assume for a start that you are seated in front of your micro, which is already plugged in (the User Instructions book is pretty good on this.) Right – that's the end of our assumptions.

Turn on your monitor – the switch is a simple push button on the front at the right. You won't see a lot happen! Then switch on the micro itself – the switch is on the right-hand side towards the back.

The first sign of life from your CPC464 should be a message similar to this:



As the message says, you've got a 64k colour computer. The "64k" is a measure of the micro's memory. In computing terms having more memory means, very roughly, being able to type more into the computer before it's full up.

Basic, as we shall see later, is the name of the language in which you give instructions to the CPC464. The remainder of the message tells you who markets the micro – Amstrad – and who wrote the Basic – Locomotive. The "Ready" means just that – the micro is ready for you to type in some information. We call it the prompt.

At the end of these initial messages – under the "R" of Ready – you should see a solid square on the screen. This is called the cursor, and it too indicates that the micro is ready for some information from you.

So let's give it some. Type in two or three letters – just the alphabet for the moment, please. It should be apparent that the cursor indicates the position at which the next character will appear on the screen.

Let's take a closer look at the keyboard. Fundamentally it is a

standard typewriter keyboard surrounded by several additional keys.

However, unlike the standard typewriter, not only do the numbers appear on the top row of keys as usual, but also on [illegible] separate "numeric keypad" to the right of the main keyboard.

Notice that the keyboard has separate keys for the letter O and the number 0. The 0, [illegible] you'll see if you try it, always appears with a diagonal line across it – whether you type it on the main keyboard or the numeric keypad.

You must keep 0 and O entirely separate. I guarantee that a lot of your early mistakes in programs will be caused by typing O instead of 0!

On the same lines, notice that there are special keys for 1. Don't [illegible] the lower case "l" for [illegible], as you have to [illegible] [illegible] typewriters – the micro won't appreciate it!

Above the numeric keypad there's [illegible] cluster of five keys – one labelled COPY, surrounded by four keys with arrows. These are simply dealt with – we're going to ignore them for the present.

Going back to the main keyboard, you'll [illegible] that [illegible] well as the letters and numbers ("alphanumerics" as the jargon has it) there [illegible] other keys, neatly picked out in colour.

Some, such as Caps Lock and Shift, you'll be familiar with from ordinary typewriters. Others, such [illegible] Ctrl and Esc, should be new to you.

Let's introduce [illegible] convention [illegible] make life easier. If I want you to press the Shift key, I'll ask you to type

[Shift]

If, [illegible] the other hand, I ask you to type

Shift

I want you to type S followed by h, followed by i and so on. So [illegible] I want you to press [illegible] single key, I'll put the name of that key in square brackets – otherwise type each character out separately.

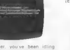
Now Enter is quite an important key – it's the blue one at the right of the main keyboard. It's also at the bottom right-hand corner of the numeric keypad.

We use Enter in [illegible] similar manner to the Return key on [illegible] electric typewriter, to ensure that the typing continues [illegible] [illegible] new line.

It's far more important than that, though. Enter not only gives you a [illegible] line, it also sends the message you have typed on that line into the computer to be acted upon.

You can't hurt the Amstrad by accidentally mistyping something – so feel free . . .

If you've been following [illegible] far, you should have a [illegible] looking something like this:



If, however, you've been idling until now, type a few letters, then press [Enter]. Odds on, you'll get [illegible] message back from the computer saying:

Don't worry about this message – you can't hurt the micro by accidentally mistyping something, [illegible] feel free to experiment.

All that *Syntax error* [illegible] is that the computer doesn't understand the words you've just sent it. You [illegible], it needs to [illegible] talked to in its own language, called Basic.

However learning Basic isn't like learning [illegible] genuinely foreign language. Basic is very similar to English, but it only uses a small set of selected English words in order to make things simpler for the computer. These selected words are called keywords.

This, by the way, [illegible] why I said it was "odds on" you'd get the *Syntax error* message when you pressed Enter. You might, by chance have hit [illegible] [illegible] Basic keyword.

For example, you [illegible] mark the end of [illegible] Basic program with the keyword *end*. The people who designed Basic could have chosen the word *finish* to do this.

Let's see if there's any difference in the words. Type *end* and press [Enter]. Then type *finish* and press [Enter]. The screen will look like this:

Notice the difference? The CPC464 accepts *end*, but not *finish*.

Admittedly, *end* doesn't accomplish very much. After all, you haven't anything in there to end, have you? But at least the micro didn't hurl the message *Syntax error* at you [illegible] it did with *finish*.

This is because *end* [illegible] a Basic keyword, while *finish* isn't.

Notice something about the *Syntax error* message – it has a capital S. So far everything we've typed in has been in lower case – at least in theory. (Remember, I said stick to the letters of the alphabet!)

I bet you've already discovered that if you hold Shift down while you're typing the letters appear in upper case. If not, try it now. It's what you'd expect, if you've ever used a typewriter.

Again it will come [illegible] no surprise to [illegible] that if you press the

key with Shift, you get the "&", while un-Shifted you get the "6".

If you press Caps Lock and release

it you'll find that the letters of the alphabet come out in upper case automatically. Also if a key has two characters (or legends as they are known), you get the upper one on the screen.

So, with the

key, you'll get "&" with Caps Lock on.

This upper case output will continue until you press Caps Lock again, then you'll go back to normal. Press it again and you're back into upper case. Once more and it's normal again. (This way of jumping between one "state" and another is called "toggling" in computer jargon.)

What do you suppose would happen if you typed in *END* instead of *end*? Try it and see. One of the nicest things about using a computer is that you don't have to speculate. If you find yourself asking "What would happen if?" or "I wonder if this would work?" you can go right ahead and try it.

As I said, you can't hurt the micro from the keyboard, so go ahead and experiment. Believe me, it's by far the best way to learn about programming.

Now you were wondering how the micro would react to *END* instead of *end*. Right then, just type in:

END [Enter]

(remembering that [Enter] means press the Enter key, don't type in E,n,t . . .). As you'll see, the micro doesn't throw it out.

In fact the Amstrad doesn't mind if you type in your keywords in upper or lower case (unlike some other, more pedantic micros that demand keywords in upper case only.)

Another thing you've probably realised is that the keys have an auto-repeat facility. This means that having pressed a key, if you keep it down the letters repeatedly print themselves out automatically.

Press Enter to get on a new line, ignoring any *Syntax error* messages. Using auto-repeat, put several letters on a line but don't press Enter. Right, you should now have a line of characters with the cursor hanging on the end of them.

Press the key marked Del – you'll see that the character on the left of the cursor is "devoured" by it.

This is one way of getting rid of, or deleting, characters (hence the Del). If you keep Del pressed down it will auto-repeat and gobble up the whole line, making a plaintive beeping sound when there's nothing left for it to eat!

You can use Del in this way to correct typing errors – just delete back to the mistake and retype.

Doing this sort of computer correcting is known as editing. There are more sophisticated ways of doing it that we'll be covering, involving the Ctrl, Copy and arrow keys. We'll leave dealing with these for another time.

By now, with all this experimenting, you'll have probably filled up a screenful of text and seen the scrolling action demonstrated. If not, press [Enter] several times in succession.

As you'll soon see, scrolling is when the top of the screen rolls up to allow more typing at the bottom.

(If you're wondering why I didn't say just hold Enter down to get the auto-repeat – it's because Enter doesn't auto-repeat.)

If you've had the patience to keep on pressing Enter until everything disappeared off the top of the screen ("scrolled off" is the jargon) you'll have a blank screen with the cursor at the bottom. There are easier ways of clearing the screen, however. Type in some garbage and then follow up with:

cls [Enter]

The screen should clear, and you'll be left with the ready prompt and the cursor at the top of the screen. If you are not convinced about the CPC464 accepting keywords in both upper and lower case, try:

CLS [Enter]

Both are equally effective.

Well, this has been just a brief examination of the keyboard. There's lots more to cover, including the Esc and Ctrl keys. For the moment though, we'll change tack – after all, it's a computer, so let's get it to compute!

Don't worry, though – this isn't going to turn into a mathematical treatise. After a brief but necessary foray into simple sums this series is thoroughly non-mathematical.

Before we start, let me give you a warning. The computer will do exactly as you tell it, but only what you tell it.

It's a very literal machine and in this respect is like my daughter on a particularly mischievous day. When asked to put on her pyjamas for bed she did exactly as she was told.

Of course, I hadn't asked her to take off her clothes first, had I? You can imagine the results.

Similar things happen with the computer. Say we wanted the micro to calculate 2 + 2. Not only do we want it to do the sum, we also want it to tell us the answer when it's finished.

We instruct the Amstrad to write things on the screen with the Basic keyword *print*. This is a relic from the days when the computer's output, as

If you keep Del pressed down it will gobble up the whole line, making a plaintive beeping sound when there's nothing left for it to eat!

it's called, was actually printed out on paper rather than a screen as it is now.

So, to [illegible] the answer to 2 + 2, type:

print 2+2 [Enter]

Note that you don't need the "=" sign as you do on a calculator. [Enter] takes care of that. Before continuing with this article, why not try [illegible] few simple addition sums? Also, try using *PRINT* instead of *print*.

Just [illegible] the micro does not allow you to use O or [illegible] for 0, [illegible] it does not let you [illegible] x or X for multiply. You must [illegible] the * symbol instead. For example try:

PRINT 4*3 [Enter]

Minus [illegible] straightforward. You'll find it sharing a key with =. Divide, however, [illegible] not ÷ but an oblique stroke /. For example 12÷4 becomes:

PRINT 12/4 [Enter]

Though this may seem a little odd at first, you have met it when dealing with fractions: 3÷4 [illegible] equivalent to the fraction $\frac{3}{4}$. Try:

PRINT 3/4 [Enter]

From now [illegible] I am going [illegible] assume you accept that before the CPC464 can act [illegible] your instructions they must be sent to it by [Enter]. I will, therefore, omit [Enter] from my examples. Make [illegible] you don't!

Before experimenting with further sums of your own devising, I'd like you to try the following sequence:

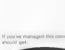
```
PRINT 2+8-3
PRINT 4*8/2
PRINT 4*8+2
PRINT 4*(8+2)
```

If you think carefully about the results you'll see that the computer interprets sequences of sums in the order you learned [illegible] school. You do whatever is inside brackets first, multiplication and division next, then finally addition and subtraction.

Now try:

```
PRINT 2/3
PRINT 1000*1000*1000
PRINT 1/100/100/100/100/100
```

If you've managed this correctly, you should get:



The point [illegible] note here is that the micro works to a limit of accuracy. For example, 2/3 is not exactly 0.666666667. The error is well under [illegible] millionth, though. Still, it must be borne in mind.

Similarly with especially large [illegible] small numbers. The computer saves space by storing them using [illegible] scientific notation called Exponent format. Here, for example, instead of printing out [illegible] [illegible] to

1000*1000*1000

as you might expect:

1000000000

it prints [illegible] the result as 1E9.

For [illegible] which stands for Exponent, you should read "multiplied by 10 [illegible] the power of". For example, 1E9 means "1 multiplied by 10 to the power of 9" which, if your maths is up to it, gives you the correct [illegible].

Similarly, the answer for:

1/100/100/100/100/100

[illegible] returned [illegible] 1E–10 which reads as "1 multiplied by 10 [illegible] the power of –10" which [illegible] 0.0000000001, the correct answer.

If you don't follow [illegible] this, don't worry. I've only covered it in this article [illegible] warn you about odd looking results to your sums which might pop up and confuse you.

Now let's try [illegible] get the computer to print out some words. Let's get it to print out Hello.

If you cast your mind back to your schooldays (and for some of us that's [illegible] awful long throw), you'll remember that when someone says something you must put what that person says with quotation marks (or quotes for short), such as, He said, "Hello".

In Basic, of course, we don't say words, we *PRINT* them, but [illegible] still surround them by quotes. We omit, however, the comma and full stop. Try:

PRINT "Hello"

The micro should print out:

Hello

Notice that the quotes are not printed. So, to get the CPC464 to print out [illegible] message on its screen, we just [illegible] *PRINT* or *print* followed by the message surrounded by quotes.

The message in quotes is called a string, [illegible] a string literal. "String" is because the micro considers them just to be a string of letters, one after the other. "Literal" is because the micro prints our literally, or exactly, what [illegible] between the quotes. So:

```
PRINT "Hello"
PRINT " Hello"
PRINT "  Hello"
```

give different outputs [illegible] in each different numbers of spaces precede the Hello.

Actually strings do not have to be words. They [illegible] be any combination of symbols, including numbers. Just keep them in quotes. Try the following:

```
[illegible] "4*3"
PRINT 4*3
```

This should convince you that the computer does print out strings – that is, what's between the quotes – literally.

When the calculation is in quotes the computer simply echoes the sum on the screen. When the calculation is not in quotes, it prints out the answer.

Experiment with printing out various messages on the screen. How long can you make them?

At the moment, the micro is responding to our commands as soon as we send them by pressing [Enter], but in a calculation or task requiring several steps this can be rather tedious.

It would be more satisfactory to give the computer a whole series of instructions that it could get on with, rather than spoon-feed it step by step.

Such a sequence of instructions is called a program, and we shall begin writing programs in next month's instalment.

Grab your paper pen and ink ...

... you're going [illegible] explore the colourful world of Amstrad graphics with the help of MICHAEL NOELS

WELCOME to the colourful world of the Amstrad CPC464, and congratulations on having such a superb machine for graphics programming.

If you've run any commercial arcade games, or typed in the programs from this issue, you've probably already seen the amazing graphics effects the CPC464 is capable of.

However because of the Amstrad's wide range of graphics and colour commands, incorporating them into your own programs can be a little difficult at first – and the User's Instructions aren't too helpful.

So here's a gentle-paced, no-nonsense introduction to the ins and outs of graphics and colour programming that you won't need [illegible] PhD in computer science to understand.

I have assumed that you know [illegible] little Basic, but don't worry if you don't – you can pick it [illegible] [illegible] you go along. And if you don't happen to have a colour monitor you can still take advantage of the techniques we'll be using.

So switch on your Amstrad, or [illegible] it [illegible] it's already on. To reset it press the Esc key while holding Ctrl and Shift down. You should see [illegible] blue screen with the familiar message appearing in yellow writing.

Let's [illegible] how many characters you can fit onto [illegible] line of the screen. Type:

0123456789

repeatedly, and you'll find that the screen has [illegible] width of exactly 40 characters.

You [illegible] actually have three basic types of screen – these [illegible] types [illegible] known as modes – all with different [illegible] widths. Enter:

```
mode 0
```

and see how many characters wide the screen is [illegible] (Incidentally, if you've got a syntax error here, you've probably missed the space out between *mode* and *0*.)

As you'll have discovered, *mode 0* has 20 characters – rather fat ones at that. Now try:

```
mode 2
```

Again the screen [illegible] cleared, but this time we get [illegible] "skinny" *Ready*. If you're up to all the typing, you'll find that you can fit 80 characters across the screen.

When you first switch on or reset you are in Mode 1. Prove it now by entering:

```
mode 1
```

As you can see, we're back to normal size, and can fit 40 characters [illegible] the screen.

So we've got three modes – 0, 1 and 2. Notice it's not 1, 2 and [illegible]. Remember, computers always start

their counting at zero, not one.

The number of characters across the screen is not the only difference between modes – they also differ in the number of colours they allow on the [illegible] at once.

Mode 0 allows 16 colours, Mode 1 four and Mode 2 permits two colours. Notice that the more colours you have the less characters you get across the screen, and vice versa.

When you think about it, it makes sense. You've only got a fixed amount of memory reserved for the screen, so if you're keeping track of a lot of colours you've not got much spare for remembering a lot of characters.

On the other hand, if you decrease the number of colours you've got to remember, there's more memory space available to keep tabs [illegible] [illegible] larger number of characters. Table I summarises it.

Mode	No. of characters	No. of colours
0	20	16
1	40	4
2	80	2

Table I: Mode characteristics

If you've been following [illegible] far, you should be in Mode 1. [illegible] not reset and we'll return to writing in yellow on [illegible] blue background, with [illegible] screen width of 40 characters.

Looking at it logically the smallest number of colours you can have in a mode is two. If you're going to see anything on the [illegible] at all you'll need [illegible] foreground colour and [illegible] background colour.

For instance, in order for you to see the writing on this page, we've

chosen black to [illegible] the foreground colour (that is, the colour that letters appear in) and white for the background (the colour of the paper).

Of course our printers could swop this round – and sometimes do – [illegible] that the letters appear in white on a black background, giving [illegible] sort of negative.

[illegible] [illegible] really wanted to go berserk we could print it in a white foreground on a white background, only you wouldn't be able to see it because of the lack of contrast. (Oddly enough, this sometimes comes in handy on the Amstrad.)

At the moment [illegible] far [illegible] the CPC464 is concerned I want you to imagine that we're writing on blue paper with [illegible] [illegible] filled with yellow coloured ink.

All right, I'll come clean – the [illegible] for the tortuous last sentence was that *pen*, *ink* and *paper* are special words as far as the Amstrad is concerned. Enter:

```
pen 2
```

and you'll see the Ready prompt as usual after [illegible] direct command, but it's changed colour. Now it appears in cyan.

If you try typing in [illegible] few characters – it doesn't matter which – they should all appear in cyan, though still on [illegible] blue background.

Next, try:

```
pen 3
```

and the writing should now appear in red. Then enter:

```
pen 1
```

and our characters will appear in yellow again.

Great! We've got three pens to write with, have we? No. We've actually got four – each filled with a different coloured ink – and, [illegible] per usual computer practice, [illegible] number them from 0 [illegible] 3.

[illegible] we want [illegible] change pens, we simply type pen followed by a space and then the number of the pen [illegible] want. So:

```
pen 3
```

puts out writing in bright red. Type:

```
pen [illegible]
```

and try [illegible] writing. You won't be able to [illegible] [illegible] thing because – if you haven't guessed yet – *pen 0* is blue so you're writing in blue ink on blue paper.

So how do [illegible] get out of it? Well, press Enter to get you on a new line, then carefully type:

```
pen 1
```

and press Enter again. You should get back to yellow writing.

If you can't manage this – and it can be really awkward typing when you can't [illegible] what you're doing – reset the machine. Now try:

```
pen 4
```

Sorry about that! You're writing in blue [illegible] blue again. Oh well, at least you know how to get out of it this time – and you know that *pen 4* is the [illegible] as *pen 0*. Reset your machine and try:

```
pen 5
```

The writing's still in yellow. So *pen 5* is the same as *pen 1*. So what about *pen 6*? Try it:

```
pen [illegible]
```

We're in cyan, the same as *pen 2*. No prizes for guessing that:

```
pen 7
```

gives you red. Let's explain: When you switch on or reset, you [illegible] in Mode 1. Now Mode 1 only allows four colours or inks on the screen at once. So when you've gone from *pen 0* to *pen 3* the CPC464 starts again by making *pen 4* equivalent to *pen 0* and so [illegible]. Similarly *pen 8* is equivalent to *pen 0*, and so on.

So in Mode 1, given a pen number, it's equivalent to the remainder left when that pen number is divided by 4 (since it's a four colour mode). Hence *pen 13* is equivalent to *pen 1*. I always imagine that the pen numbers [illegible] "wrapping round" to the start again.

If the above maths has you foxed don't worry. If you don't try anything fancy, and stick to the numbers 0 to 3 for your Mode 1 pens, you'll be all right. Table II summarises the colours

Pen number	Colour
0	**Bright blue**
1	**Bright yellow**
2	**Bright cyan**
3	**Bright red**

Table II: Default colours in Mode 1

associated with the pen numbers.

Now run Program I, which illustrates the different colours available. You could add the following lines to illustrate *pen 0*:

```
[illegible] pen [illegible]
100 PRINT "This is in pen [illegible]"
```

but, of course, you wouldn't see it.

```
[illegible] REM PROGRAM I
20 MODE 1
30 PEN 1
40 PRINT "This is in pen 1"
50 PEN 2
[illegible] PRINT "This is in pen 2"
70 PEN 3
[illegible] PRINT "This is in pen 3"
```

What would happen if we ran it in Mode 2? Change line 20 [illegible]:

```
[illegible] mode 2
```

and see.

What happened to *pen 2*, and why's *pen 3* yellow? It used to be red! Well Mode 2 is a two colour mode. *Pen 0* gives us bright blue, *pen 1* gives us bright yellow – then we've run out of available colours, so we start again as we did when [illegible] ran out of colours in Mode 1 (only then there [illegible] four available).

So *pen 2* wraps round to blue and disappears against the blue background, while *pen 3* becomes yellow, and [illegible] on. Table III shows the colours associated with the pen numbers in Mode 2.

If we change line 20 to:

```
20 mode [illegible]
```

there seems to be little difference from when you [illegible] it in Mode 1, save for the fatter characters. Don't forget, though, this is [illegible] 16 colour mode – our pens should go from 0 to 15.

Program II illustrates the idea – showing all 16 colours – including the rather natty flashing colours of pens 14 and 15. Table IV shows the

```
10 REN [illegible] II
20 MODE [illegible]
30 FOR colour = [illegible] TO 15
[illegible] PEN colour
[illegible] PRINT "This is colour ";colour
[illegible] NEXT colour
```

Pen number	Colour
0	Bright blue
1	Bright yellow

Table III: Default pen colours in Mode 2

colours associated with the pen numbers in Mode 0.

Try changing line 20 to give Modes 1 and 2 and you'll [illegible] how in modes with less colours the pen numbers wrap around. If you now enter *pen 16*:

```
Improper argument
```

will be hurled back [illegible] you. The CPC464 knows that the biggest pen number it can possibly have is 15, [illegible] it throws *pen 16* out. In Modes 1 and 2, as we've seen, it wraps the pen numbers round, but it still rejects numbers over 15.

So far all our work has been done [illegible] [illegible] nice blue background, but [illegible] aren't restricted to this. Let's investigate.

Reset your micro so you are back in Mode 1. Now so far we've been writing with pens filled with different coloured inks – on blue paper. Enter this:

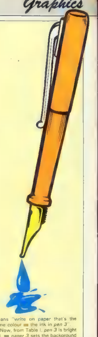
```
paper 3
```

All of a sudden Ready appears on red paper. That is, the letters still appear in yellow, but on [illegible] red background. You see:

```
paper 3
```

Pen number	Colour
0	Bright blue
1	Bright yellow
2	Bright cyan
3	Bright red
4	Bright white
5	Black
6	Bright blue
7	Bright magenta
8	Cyan
9	Yellow
10	Pastel blue
11	Pink
12	Bright green
13	Pastel green
14	Flashing blue/ bright yellow
15	Flashing pink/ sky blue

Table IV: Default pen colours in Mode 0



means "write on paper that's the same colour [illegible] the ink in *pen 3*".

Now, from Table I, *pen 3* is bright red. [illegible] *paper 3* sets the background to bright red. Type in some characters of your own if you don't believe me. Next try:

```
paper 2
```

The ink in *pen 2* is bright cyan, so our writing now appears on cyan paper. I find this terribly difficult to read, so let's make it clearer by

changing our foreground colour to red. Remember how? It's:

```
pen 3
```

Paper colour is really quite easy to [illegible] – it works just [illegible] pen does, and follows the same restrictions as to mode. Just bear in mind: *paper n* [illegible] the background colour is that of the ink in *pen n*.

Notice that so far only the background of the characters you've typed has been in the new paper – the rest of the line stays in the old paper. When you've reached the bottom line, however, and the new line scrolls up, the whole of that line will be in the new paper.

After all, it's got to be in something, and as it's brand spanking new we may [illegible] well have it in the new paper.

There is a quicker way to get the screen in the new background colour. Enter:

```
paper 1 : cls
```

and the screen will clear to a yellow background *(paper 1)* with writing in the red foreground colour (we're still in *pen 3*).

You'll also notice something else if you haven't already – our yellow paper is surrounded by a blue border. You haven't noticed it before because our background's always been blue, matching the border.

We'll see later how you can change the border's colour. In fact there's not much else you can do with it – we can't actually write anything there . . .

Before we continue, have a look at Program III, which illustrates how

```
10 REM PROGRAM III
20 MODE 0
30 FOR background = 0 TO 15
40 PAPER background
50 CLS
60 PRINT"This is paper "; background
70 PRINT:PRINT
80 FOR colour = 0 TO 15
90 PEN colour
100 PRINT "This is colour ";colour
110 NEXT colour
120 PRINT:PRINT:PRINT"Press any key"
130 delay$ = INKEY$ : IF delay$="" TH
EN GOTO 130
140 NEXT background
150 PAPER 0 : [illegible] 1
```

the various pen and paper combinations work.

So far we've only seen 16 colours. However, when you bought your Amstrad you were promised 27. What's happened to them?

Program IV shows where they've been hiding. It successively steps the border through all 27 colours of inks, as they are known.

```
10 REM PROGRAM IV
20 MODE 1
30 FOR colour = 0 TO 26
40 BORDER colour
50 LOCATE 16,12
60 PRINT "Border "; colour
70 FOR delay = 0 TO 500 :NEXT delay
80 NEXT colour
```

As you'll have guessed *border* is the command that changes the colour of the border – you simply follow [illegible] with a space and the number of the colour you want the surround to be.

But beware, these numbers won't appear to have anything to do with the numbers you've been using for pen and paper. For example:

```
border 0
```

The border turns black – not blue [illegible] you would expect from pen 0 and paper 0.

This is a very important point – the numbers used with pen don't label colours – they label pens, which just happen to [illegible] filled with "coloured inks".

Just because *pen 3* has so far always given us red [illegible] Mode 1, it doesn't have to. It's just that, at the moment, *pen 3* happens to be filled with red ink.

Later on I'll show you how to fill [illegible] pen with, say, blue ink – in fact any coloured ink from our "palette" of 27 colours.

So the 3 in *pen 3* labels the pen, not the colour of the ink it is filled with. As in Mode 1 we're allowed four pens, and hence four corresponding papers. We [illegible] fill these pens with any four of the 27 – a sort of "perm any 4 from 27". In fact you can fill all four pens with the same colour if you want.

Much the same holds for the other modes, with their different number of pens.

Now the micro needs some way of referring to each of the 27 available colours. It could, of course, do it in

words – orange, bright red and so on.

Being a computer, it prefers to give the various coloured inks reference numbers, [illegible] shown in Table V.

As you can see, ink 0 is black and as the border command [illegible] the ink number NOT the pen number:

```
border 0
```

turns the border black – and leaves the screen entirely alone. You must [illegible] pen [illegible] paper to affect the screen.

Next month I'll show you how to fill the pens with any ink you care to choose. For now, though, it's probably best if you just use the inks that the pens are "supplied" with when you switch on or reset – the default inks as they are known. Tables II, III and IV show them for each mode.

That should keep you busy enough until our next issue!

Ink number	Colour
0	Black
1	Blue
2	Bright blue
3	Red
4	Magenta
5	Mauve
6	Bright red
7	Purple
8	Bright magenta
9	Green
10	Cyan
11	Sky blue
12	Yellow
13	White
14	Pastel blue
15	Orange
16	Pink
17	Pastel magenta
18	Bright green
19	Sea green
20	Bright cyan
21	Lime green
22	Pastel green
23	Pastel cyan
24	Bright yellow
25	Pastel yellow
26	Bright white

Table V: Ink colours

TAKE A TOUR ROUND YOUR MICRO'S SOUND

First in an informative series by NIGEL PETERS

MOST micros nowadays have the ability to make sounds. The simplest way to get your CPC464 to make a [illegible] is [illegible] enter:

```
PRINT CHR$(7)
```

and press Enter. The trouble is that this solitary beep isn't all that exciting.

To enable you to create rather more interesting sounds the Amstrad has three very sophisticated commands – SOUND, ENT and ENV. However, the more sophisticated the command the more difficult it is to use, at least at first.

So over the next few months I will be covering the commands in detail, building up a step-by-step guide to making noises on the Amstrad CPC464. It will be [illegible] practical course with lots of examples. [illegible] make sure that your micro is switched [illegible] and ready for use while you read the articles.

And don't take everything I say on trust – try [illegible] out. The SOUND command is [illegible] big subject and only by trying things out for yourself will you grasp it in its entirety.

Having said all that, let's make [illegible] sound. Enter:

```
SOUND 1,200,100,7
```

and feel yourself thrill to the exciting [illegible] of your Amstrad.

Well, maybe I am overdoing it a bit. It's [illegible] the most exciting sound in the world. Still once you understand how that SOUND command works you'll have gone a long way to conquering Amstrad sound.

The simplest form of the SOUND command is followed by four parameters. Don't be put off by parameters, they [illegible] just the numbers that follow the SOUND command and affect [illegible] way it works.

The parameters [illegible] the last sound [illegible] made [illegible] 1, 200, 100 and 7. The first number (1) selects the channel that the note will be played on, the second (200) tells the micro what pitch it will be. The third parameter (100) determines how long that note will last while the last figure (7) decides on the volume it will be played at.

By entering:

```
[illegible] 1,200,100,7
```

[illegible] produce [illegible] note [illegible] channel 1 which will last for [illegible] second, be played at full volume and be fairly high pitched.

Don't worry [illegible] you don't fully understand all these terms, we'll be dealing with them later in this article.

The basic structure of the SOUND command is:

```
SOUND channel,pitch,duration,volume
```

that is, the Basic keyword SOUND followed by four parameters which specify the particular sound wanted.

Have a go at [illegible] few more sounds by entering:

```
[illegible] 2,600,20,4
```

or:

```
SOUND 4,20,50,3
```

or be adventurous and make up your own.

As you'll see (and hear) from the above, the values following the SOUND command can vary and as they vary so do the notes produced. However you can't just pick any old parameter, you can only choose between certain limits.

Take the channel parameter, the first number that comes after the SOUND command. This is used to decide which of the CPC464's sound channels is to be used. The Amstrad has three such channels, and each can only play one note at [illegible] time.

However by having all three channels working at the same time you can have the micro producing three different notes simultaneously. The channels are known [illegible] channels A, [illegible] and C and the parameters that

select them are 1, 2 and 4 respectively.

Program I shows these parameters in action. First of all it plays [redacted] note [redacted] channel A (parameter 1), then waits for you to press [redacted] key. The next note [redacted] played [redacted] channel B as the parameter following the SOUND command [redacted] the figure 2.

```
10 [redacted] I
[redacted] Channel A
[redacted] SOUND 1,478,100,7
[redacted] WHILE INKEY$="":WEND
[redacted] REM Channel B
[redacted] SOUND 2,379,100,7
70 WHILE INKEY$="":WEND
80 [redacted] Channel C
[redacted] SOUND 4,301,100,7
```

Program I

When this note is over and you press another key, the last note, with channel parameter 4, plays on channel C.

Of course these notes were played [redacted] after another. I hope the more sceptical of you will be thinking: "How do I know there [redacted] three channels? There could [redacted] just one channel with the notes playing [redacted] it, [redacted] after another".

If you do have such doubts, have a go at Program II which plays the same notes at the same time, [redacted] different channels. Not exactly Mozart is it? Still, it should have convinced you about the three channels.

Now let's move on from the channel parameter – *we'll be back to it in [redacted] later article* – and look at the pitch parameter that follows it. As you might guess, this controls the pitch of the note produced by the SOUND command.

The pitch of [redacted] note [redacted] how high or low it is. The higher the note – like a soprano or a molested cat – the higher the pitch [redacted]. The lower the note

```
[redacted] REM PROGRAM II
20 [redacted] Channel A
30 [redacted] 1,478,100,7
[redacted] REM Channel B
[redacted] 2,379,100,7
[redacted] Channel c
[redacted] 4,301,100,7
```

Program II

– like [redacted] bass [redacted] foghorn – the lower the pitch.

Run Program III and you'll hear [redacted] note that changes pitch, going lower and lower. Press ESC a couple of times when you've had enough.

What's happened is that each time round the FOR ... NEXT loop the SOUND command of line 30 makes a note. However each time round the loop the value of the pitch parameter, *pitch*, varies. At first it's 32, then it's 33, then 34 and [redacted] on. As the pitch parameter varies so does [redacted] note produced.

```
[redacted] PROGRAM III
[redacted] pitch=32 TO [redacted]
[redacted] SOUND 1,pitch,100,7
[redacted] NEXT pitch
```

Program III

You might have noticed that as the pitch parameter gets larger in value, the note gets lower in pitch (the sound gets deeper). This means that [redacted] pitch parameter of 100 produces a much lower note than [redacted] pitch parameter of [redacted].

Incidentally if you did [redacted] ESC to get out [redacted] the previous program you might have left a few [redacted] in the micro. These may cause the following programs to sound odd (or, rather, odder) so get rid of them by entering:

```
SOUND 129,100,0,0
```

I know that [redacted] channel parameter looks wrong, but all will be explained in a later article. Just [redacted] for garbage collection for the time being.

To return [redacted] the pitch parameter, Program IV uses [redacted] step of –4 to decrease the pitch parameter each time round the FOR ... NEXT loop. The result is a note that increases in pitch.

The pitch parameter [redacted] take values that range from 0 to 4095 though it does get [redacted] ragged at the extremes of its range. The numbers have to be integers (whole numbers). If you [redacted] number with [redacted] decimal part for the pitch parameter the CPC just ignores the decimal.

Don't just believe [redacted], try it for yourself. Is there [redacted] difference between:

```
SOUND 1,200.5,100,7
```

and

```
SOUND 1,200,100,7
```

that you can detect?

For those with musical pretentions

```
10 REM [redacted] IV
[redacted] pitch= 478 TO [redacted] STEP -4
[redacted] SOUND 1,pitch,100,7
[redacted] pitch
```

Program IV

Appendix VII of the User Instructions gives the full range of pitch parameters with their corresponding musical notes. They call the pitch parameter the tone period, but they're the same thing.

Table I shows 24 of these notes and their pitch parameter values.

Program V plays [redacted] selection of these notes in order, reading the values of *pitch* from the DATA statements of line 60.

```
10 REM [redacted]
20 FOR note=1 [redacted] 12
30 [redacted] pitch
40 [redacted] 1,pitch,100,7
50 NEXT note
60 DATA 119,113,106,100,95,89
70 [redacted] 84,80,75,71,67,63
```

Program V

You might notice that this series of notes has a [redacted] "complete" feel about [redacted] than the previous [redacted] we've had. We'll come back to this in a later article.

For the moment let's press [redacted] to the next parameter which determines

[redacted]	Pitch parameter	[redacted]	Pitch parameter
C	239	F#	[redacted]
C#	225	[redacted]	[redacted]
D	213	G#	75
D#	201	A	71
E	190	A#	67
[redacted]	179	[redacted]	63
[redacted]	169	C	[redacted]
G	159	[redacted]	[redacted]
G#	150	D	53
A	142	D#	[redacted]
[redacted]	134	E	47
B	127	[redacted]	[redacted]
C	119	F#	42
C#	113	G	[redacted]
D	106	G#	[redacted]
D#	[redacted]	A	[redacted]
E	95	A#	[redacted]
F	89	B	[redacted]

Table I

how long the note is going to last.

This third parameter, controlling duration, can have values from -32768 to 32767 but for the present we'll just use the values 1 to 32767. This number represents the number of 1/100ths of a second the note is to last.

So if the duration parameter is 100 the note should last for [redacted] second. If the duration parameter is 1000 it should last for 10 seconds. Enter:

```
[redacted] 1,200,300,7
```

and you'll find the note lasts for three seconds.

Program VI, a variant of the previous one, [redacted] the familiar FOR ... NEXT loop to vary the duration parameter via the variable *duration*.

```
10 [redacted] VI
20 FOR duration=10 TO 120 STEP 10
[redacted] READ pitch
[redacted] 1,pitch,duration,7
50 NEXT duration
[redacted] 119,113,106,100,95,89
[redacted] DATA 84,80,75,71,67,63
```

Program VI

As the notes change [redacted] pitch each time round the loop they also increase in length from 1/10th of [redacted] second to 1.2 seconds. This gives a sort of slowing down effect.

Program VII decreases the duration parameter each time round the loop so giving the opposite effect.

The final parameter we're going to deal with is the volume parameter. This, as is obvious from the name, decides how loud the note is going to be and can have whole number

```
10 REM PROGRAM VII
[redacted] duration=120 TO [redacted] STEP -10
[redacted] READ pitch
[redacted] SOUND 1,pitch,duration,7
50 NEXT duration
[redacted] DATA 119,113,106,100,95,89
70 [redacted] 84,80,75,71,67,63
```

Program VII

values from 0 to 7. [redacted] the parameter is 1 the note is at its quietest. The parameter [redacted] is a little louder and [redacted] on to maximum volume when the volume parameter is equal to 7. Program VIII shows them [redacted] action.

```
10 REM PROGRAM VIII
[redacted] FOR volume=1 TO 7
[redacted] SOUND 1,100,100,volume
40 SOUND 1,100,100,0
[redacted] NEXT volume
[redacted] DATA 119,113,106,100,95,89
70 DATA 84,80,75,71,67,63
```

Program VIII

The main part of the program is easy enough to follow. As the FOR ... NEXT loop cycles the volume parameter increases and the note gets louder. But what of line 40 which has [redacted] SOUND command with [redacted] volume parameter of 0? What's that doing there?

The answer is that it's there to provide an interval of no sound between the other notes. When the volume parameter has [redacted] value of 0 it means that the sound has no loudness at all. You can't hear it.

However the note still plays (albeit silently) for the full length of the duration parameter. This means that line 40 produces one second silences between the notes.

Although using a 0 volume parameter to produce a note that you can't hear seems a little strange at first sight it is useful. You can use it to produce the rests that are often found in tunes. Also it makes things a lot clearer at times. Leave out line 40 in the last program and you'll see what I mean.

And that's the end of our tour of the four SOUND parameters, except for [redacted] thing. Can you explain why the notes produced by:

```
[redacted] 1,200,20,4
```

and

```
SOUND 1,200
```

sound exactly the same?

The reason is that if we don't supply the SOUND command with a duration [redacted] volume parameter it provides them for itself. The default duration parameter [redacted] 20 and the default volume is 4, hence the two SOUND commands above produce the same note.

And that's it for this month. Table II sums up what we've covered so far, while Program IX makes use of it to produce [redacted] familiar tune.

Why not write [redacted] few simple tunes yourself?

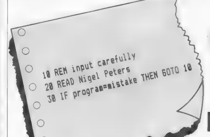
```
[redacted] PROGRAM IX
[redacted] SOUND 1,239,100,7
[redacted] 1,213,100,7
40 SOUND 1,268,100,7
50 SOUND 1,536,100,7
60 SOUND 1,358,100,7
```

Program IX

- ***Next month:*** *More explanation of the sound capabilities of the Amstrad CPC464.*

	Channel	Pitch	Duration	Volume
range	1 = A 2 = B 4 = C	0 to 4095	1 to 32767	0 to 7
default	none	none	20	[redacted]

Table II: Parameter ranges for SOUND command



How to avoid those listings loopholes

ONE of the good things about this magazine is that it's going to be full of listings. Long ones, short [illegible], simple [illegible] and hard ones – they'll all be there, waiting for you to type them into your CPC464.

This, however, is sometimes easier said than done. If you're not careful or are just plain unlucky, typing in listings can be extremely frustrating.

I'm sure we've all had the experience at one time [illegible] another of typing something in and finding that it doesn't work. And no matter how hard we try, the fault is untraceable. [illegible] you haven't had this happen to you yet, touch wood, quickly.

Eventually we give up in disgust and say that the listing [illegible] wrong. Usually in fairly strong language.

To be fair, this can be the case, though it very rarely is. Occasionally a part of [illegible] listing won't print out properly or a space will appear where there wasn't one in the first place. It's even been known for part of a listing to "fall off" [illegible] page.

However this doesn't happen all that often (though once is one time too many). The sad fact is that if the program won't work you've almost certainly made [illegible] error typing the listing [illegible] your [illegible]. There's no one else to blame.

I speak from experienc[illegible] at one time or another I've made every [illegible] possible (and some that I'm [illegible] aren't). Even [illegible] after hundreds of hours at a keyboard, when I should know better I still end [illegible] looking at a listing convinced that I've not made a mistake. But, in my heart of hearts I know that the odds are that I have.

So from my vast experience of typing in listings wrongly I'll give you [illegible] few hints on how to stop those errors creeping in.

The first thing is *don't [illegible] carried away*. It's all too easy to become obsessed with a listing that's gone wrong, struggling with it for hour after hour, eventually losing your temper.

If you don't spot the mistake in the first quarter of an hour then give up for a little while. Go and have a cup of coffee [illegible] something. See if the family still recognises you. It's surprising how often [illegible] [illegible] the mistake [illegible] soon as you try again after [illegible] rest.

So don't get too involved. It doesn't help. And remember, [illegible] do it for fun, our lives don't depend [illegible] it.

One of the daftest mistakes you can make [illegible] to leave [illegible] complete line out. Of course the program won't work properly if it isn't all there, but lots of times [illegible] expect it to.

Often we just miss the line, skipping it by accident as we read the listing. Alternatively, instead of typing in two lines, 40 and 50, we type in the first line [illegible] line 40 and then number the next one as 40. The second line overwrites the first and [illegible] we have a missing line. To make it worse, the line numbered 40 is actually the one that should be numbered 50. It's amazing how difficult this can be to spot.

And the trouble is that the Amstrad can't tell you that "line 50 is missing" because (unless it's used in [illegible] GOTO or GOSUB) it doesn't know that it should be there. So the error message that it comes up with isn't all that helpful, often pointing to some other part of the program that depended on the missing line.

Suppose line 50 was something like:

```
cowboys=8
```

that is, it gives the variable *cowboys* [illegible] value of eight. Now if you leave this out the micro will either use an earlier value of *cowboys*, which might be completely wrong, or assume, since [illegible]

hasn't come across it before, that it's equal to zero.

Either way it won't tell you that there's [illegible] line missing. You have to hunt through and find it for yourself.

The moral is that when you're typing in a listing, type in all of it. And type it in exactly. There's no point in changing the programmer's code and then feeling aggrieved when it doesn't work.

Yet people do this, correcting "mistakes" that they find in listings and then wondering why the program grinds to [illegible] halt 20 lines later.

So type in the listings exactly - and watch your spelling, because the Amstrad [illegible] very sensitive about such things. It can only understand a few words, the Basic keywords such as PRINT, LIST and LET.

Misspell these and you're in trouble, as Program I shows. Happily, you [illegible] told if you've done this, so it's not too hard to track down.

```
[illegible] REM Program I
[illegible] LET indians=5
[illegible] PRNT "There are ";indians;" indians"
```

Of course, *you* wouldn't do anything so daft, would you?

And it's not just keywords where spelling is important. Misspell [illegible] variable and the micro can easily get confused. Look at Program II.

```
10 [illegible] Program II
[illegible] LET indians=5
[illegible] PRINT "There are ";indans;" indians"
```

The micro comes across *indans*, can't find any reference to it and gives it a value of zero. In this case it's immediately apparent that there's been [illegible] error, but in long listings these "ghost" variables can be very difficult to spot.

So if you want a listing to work, you have to type it in correctly. This is easier said than done, [illegible] it's amazing how your mind can wander as you're doing it.

I've found that it makes life [illegible] lot easier if I enter [illegible] long listing in three or four sessions rather than one, saving it to tape between times.

Incidentally, I recommend that you SAVE your program every 20 or 30 lines. This is in case you lose your listings by an accident such as a power cut or [illegible] meddlesome younger brother. You might lose all your work from the Amstrad's memory but you will have the best part of it [illegible] tape.

So far we've looked at what can go wrong if we mispell Basic keywords and variable names. Now let's look at [illegible] classic error that I'm [illegible] everyone has made at [illegible] time or another. This is caused by mistaking a number for a letter and vice versa.

Probably [illegible] most common, and the hardest to detect, is confusion between the numeral "0" and the lower case letter "o".

On poor quality listings these look very much the same, and [illegible] with clear listings it's all too easy to press one key in mistake for the other. Even with [illegible] slash across the 0 this can happen.

The Amstrad won't like it and the program will almost certainly grind to a halt. Try out Program III and [illegible] what happens.

```
[illegible] REM PROGRAM III
20 FOR loop=1o TO 20
30 PRINT "0 and [illegible] are mixed up"
40 NEXT loop
```

The [illegible] kind of confusion can [illegible] between the lower case letter "l" and the number 1. They look fairly similar and on typewriters they [illegible] often the [illegible] key - but not [illegible] the Amstrad. Confuse the two and you're asking for trouble – trouble that's very difficult to spot and sort out.

Another pair of lookalikes to be wary of [illegible] the minus sign "–" and the underline "_" which are often confused. You can't do a subtraction with the underline sign, though it's amazing how many times you try!

Now let's move on from the problems caused by similarities between letters and numbers to problems caused by getting the punctuation wrong.

In particular it's all too easy to confuse the fullstop ".", the semi-colon ";", the colon ":", and the comma ",". The trouble is that these can look very similar on listings and mistakes are easily made.

This can cause all sorts of problems from having displays in the wrong place or in the wrong colour to having the program crash without an error message.

Also beware of putting in commas in numbers. You may write 20,000 but the CPC won't like it if you type it in. It prefers the number in the form 20000 – without the comma.

Another problem caused by mixed up punctuation marks comes in data lists, where the items are separated by commas. For example, what should be:

```
DATA 3,3.5,4,4,5
```

might be typed in as:

```
DATA 3,3,5,4,4,5
```

The decimal point in 3.5 has become a comma. This will result in the wrong data being read and the program will run incorrectly, if at all.

What's particularly annoying is that the CPC won't tell you that there is now one item too many in the list. It only displays an error message when it tries to READ from a list with too few items.

This can be very tricky to spot and if you get some weird happenings in [illegible] program it's always a good idea to check that the DATA lines are spot on.

The moral is to be very careful with punctuation marks in listings. They may not mean a lot to you, but they do to the micro. Get one wrong and it can be the devil of a job to find it and remedy it.

From punctuation marks let's turn our attention to spaces and the problem caused by [illegible] lack of them.

As you know, Basic uses [illegible] special set of words, the keywords, for special purposes. The Amstrad allows us to enter them in lower or uppercase letters. However when we list them they appear as capitals.

These keywords must have [illegible] space on either side of them or else the CPC thinks that it is just part of a variable name.

Program IV shows what happens if

the space after the keyword PRINT [illegible] missed out.

```
10 [illegible] PROGRAM IV
[illegible] x=5
[illegible] PRINTx
```

Alternatively, you [illegible]n leave out the space before the keyword [illegible] in Program V. So the lesson is always to leave spaces before and after Basic keywords.

```
10 REM PROGRAM V
20 x=5
[illegible] IF x>2 [illegible]
```

You may have noticed from the above that the CPC allows you to use keywords in variable names. It's perfectly possible to have a line like:

```
10 thenprint=5
```

which gives the variable *thenprint* the value of [illegible]. My advice is to avoid using variables which contain keywords, it's fraught with difficulties.

With all these possible mistakes it's [illegible] wonder that any listings [illegible] get typed in correctly! And on top of [illegible] this there are two more simple errors that can also be made.

The first is to use the AUTO command to make the micro print out the line numbers, then forget that you're using [illegible] and type in the line number again.

A line number like 20 20 is a [illegible] giveaway that you've made this mistake. I've done it so often now that I never use AUTO. The time I've "saved" not typing in line numbers is used up correcting my mistakes.

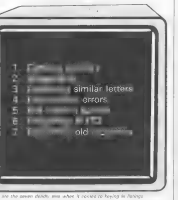
Incidentally, you'll find that if you make this mistake 20 20 is treated [illegible] line 20. Program VI illustrates this.

```
10 [illegible] PROGRAM VI
[illegible] 20LET indians=8
30 [illegible] "You've forgot about AUTO."
```

The final error is really stupid but, sadly, all too easy to make. It's done by forgetting that you've already got [illegible] program in the Amstrad's memory. You then type in your listing and, if the line numbers of the two programs don't match, they get mixed up.

The last line of Program VII is obviously left [illegible] from an earlier program and shouldn't [illegible] there [illegible]

```
10 REM PROGRAM VII
[illegible] LET indians=8
[illegible] LET cowboys=4
[illegible] IF door$=open$ THEN [illegible] "Enter at
your peril"
```



These are the seven deadly [illegible] when it comes to keying [illegible] listings

And that brings us to the end of [illegible] survey of the mistakes that [illegible] be made when typing [illegible] listings. Don't worry, you won't make all of them [illegible] the [illegible] program, though I'm sure everyone will make them at one time or another.

Now that you know what they [illegible] they should be [illegible] to find and correct. One point [illegible] bear in mind is that when you do find a mistake in a line don't think that it's the only one.

If you made one error then your concentration was lacking at that point and there may be another lurking about on the same line [illegible] thereabouts. It's amazing how often they come in pairs.

And beware of making another mistake when you "correct" [illegible] line. It's very [illegible] to introduce another error as you correct the first one – ask any printer (not ours, of course!)

All in all, it just comes down to looking carefully at what you're typing, both on the page and on the screen. If you spend all your time peering [illegible] the keyboard (as most beginner typists do) and don't look [illegible] the screen to check your work, mistakes will abound. This seems obvious, yet it's easy to become hypnotised by the keys, typing rapidly but not checking as you go.

So pay attention to what you're doing when you type in listings. A little time spent checking as you actually enter the lines saves a lot of time later.

TRAPPER

TRAPPER is a quick action game in which your aim is to trap the evil [redacted]ze monster [redacted] fast [redacted] you can. The only way to do it is to corner him so that he cannot move up, down, left or right.

The maze monster is very intelligent and will escape from the most awkward situations. So be warned. You take the part of a hunter who is controlled by the following keys:

Z – Left
X – Right
/ – Down
] – Up

The monster is afraid of everyone, [redacted] colliding with him sends him off in a totally different direction. This can be used to your advantage, [redacted] you will see when you play the game.

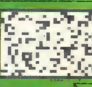
Happy trapping!

```
10 RANDOMIZE TIME
20 [redacted] Trapper
30 REM By Kevin Edwards
[redacted] REM (c)
[redacted] MODE 1
[redacted] 0:PEN 1
70 LOCATE 12,8
80 PRINT"T r a p p e r !"
90 LOCATE 1,12
100 INPUT"Enter [redacted] type, 1=simple 2
=complex ",level
110 IF level<1 [redacted] level>2 THEN SOUND
1,200,50:GOTO [redacted]
120 CLS
130 mon=224:me=249
140 block$=STRING$(40,CHR$(143))
150 LOCATE 1,1:PRINT block$
160 LOCATE 1,24:PRINT block$
```

```
170 [redacted] loop=1 TO 23
180 LOCATE 40,loop:PRINT CHR$(143);CH
R$(143)
190 NEXT
[redacted] FOR loop=1 TO level*140
210 LOCATE INT(RND(1)*38)+2,INT(RND(1
)*22)+2
220 PRINT CHR$(143)
[redacted] 1,400-loop,1
240 NEXT
250 myx=10:myy=10
260 monx=20:mony=10
[redacted] 680:GOSUB 700
280 previous=TIME
290 type=INT(RND(1)*4)+1
300 ON type [redacted] 310,320,330,340
310 rx=1:ry=0:GOTO [redacted]
```

```
320 rx=-1:ry=0:GOTO [redacted]
330 rx=0:ry=1:GOTO [redacted]
[redacted] rx=0:ry=-1
350 pix=TEST(16*(monx+rx-1),16*(25-(m
ony+ry))+8)
360 IF pix<>0 THEN [redacted] 1,200,1:GOTO
 290
370 mon=32:GOSUB 700
[redacted] monx=monx+rx:mony=mony+ry
390 mon=224:GOSUB 700
400 PEN 2:LOCATE 16,25:PRINT"time = "
;INT((TIME-previous)/300);
410 mrx=0:mry=0
420 IF INKEY(71)=0 THEN mrx=-1:GOTO 4
[redacted]
430 [redacted] INKEY(63)=0 THEN mrx=1:GOTO 46
0
```

By KEVIN EDWARDS



VARIABLES

block$	40 block characters used to draw the outside of the maze.
fin	Time taken to trap the monster.
level	Maze level. 1=simple 2=complex.
loop	General purpose loop variable.
me	Hunter's graphic character
mon	Maze monster's graphic character.
monx	Monster's X coordinate.
mony	Monster's Y coordinate.
myx	Hunter's X coordinate.
myy	Hunter's Y coordinate.
pix	Pixel colour of screen position last tested.
previous	Time when the current screen started.
rx	Monster's relative change in the X direction.
ry	Monster's relative change in the Y direction.
type	Indicates monster's present direction.

Variables not listed are general purpose ones.

```
440 IF INKEY(22)=0 THEN [illegible]y=1:GOTO 46
0
[illegible] IF INKEY(19)=0 [illegible] mry=-1
460 [illegible] mrx=0 [illegible] mry=0 THEN 300
470 pix=TEST(16*(myx+mrx-1),16*(25-(m
yy+mry))+8)
480 IF pix<>0 THEN 300
490 me=32:GOSUB 680
500 myx=myx+mrx:myy=myy+mry
510 me=249:GOSUB 680
520 w=TEST(16*(monx-2),16*(25-mony)+8
)
[illegible] x=TEST(16*monx,16*(25-mony)+8)
[illegible] y=TEST(16*(monx-1),16*(26-mony)+8
)
550 z=TEST(16*(monx-1),16*(24-mony)+8
)
560 [illegible] w<>0 [illegible] x<>0 [illegible] y<>0 [illegible] z<>
0 THEN 570 ELSE 300
570 fin=(TIME-previous)/300
580 [illegible] loop=1 TO [illegible] ],loop,1:S
OUND 2,50-loop,1:NEXT
[illegible] CLS:PEN 1:LOCATE 10,12
600 PRINT"You've trapped him !"
610 LOCATE 7,14
620 PRINT"In only ";INT(fin*100)/100;
" seconds !"
630 LOCATE 10,16
[illegible] PRINT"Another [illegible] (Y/N) ?"
[illegible] a$=UPPER$(INKEY$)
660 IF a$="" [illegible] (a$<>"Y" [illegible] a$<>"N")
[illegible]
670 IF a$="Y" THEN GOTO 50 ELSE PRINT
:PRINT:PRINT"Bye bye !!":END
[illegible] me=32 THEN PEN 1 ELSE PEN 2
690 LOCATE myx,myy:PRINT CHR$(me):RET
URN
700 [illegible] mon=32 THEN PEN 1 ELSE PEN 3
710 LOCATE monx,mony:PRINT CHR$(mon):
[illegible]
```

Give your fingers a rest ...

All the listings from this month's [illegible] are available on cassette. See our special offer on Page 55.

Ready Reference: Graphics

Get the facts [redacted] your fingertips with the first of our ready reference charts

pen/paper	ink	colour
0	1	Bright Blue
1	24	Bright Yellow
2	20	Bright Cyan
3	6	Bright [illegible]
4	26	Bright White
5	0	Black
6	2	Bright blue
7	8	Bright Magenta
8	10	Cyan
9	12	Yellow
10	14	Pastel Blue
11	16	Pink
12	18	Bright Green
13	22	Pastel [illegible]
14	1,24	Flashing Blue / Bright Yellow
15	16,11	Flashing Pink [illegible] Sky Blue

Mode 0 defaults

Ink	Colour	Ink	Colour
0	Black	14	Pastel Blue
1	Blue	15	Orange
2	Bright Blue	16	Pink
3	Red	17	Pastel Magenta
4	Magenta	18	Bright Green
5	Mauve	19	Sea Green
6	Bright Red	20	Bright Cyan
7	Purple	21	Lime Green
8	Bright Magenta	22	Pastel Green
9	Green	23	Pastel Cyan
10	Cyan	24	Bright Yellow
11	Sky Blue	25	Pastel Yellow
12	Yellow	26	Bright White
13	White		

Ink colours

pen/paper	ink	colour
0	1	Bright Bl[illegible]
1	24	Bright Yellow
2	20	Bright Cyan
3	6	Bright Red

Mode 1 defaults

pen/paper	ink	colour
0	1	Bright Blue
1	24	Bright Yellow

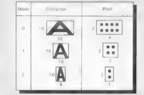
Mode 2 defaults

Mode	Number of characters	Number of colours
0	20	16
1	40	4
2	80	2

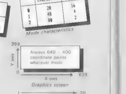
Mode characteristics

Mode	graphics coordinates		pixels	
	x axis	y axis	x axis	y axis
0	[illegible]	400	160	200
1	[illegible]	[illegible]	320	200
2	[illegible]	400	640	200

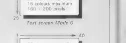
Graphics coordinates and pixels



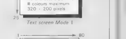
Characters and pixels measured in screen coordinates



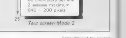
Graphics screen



Text screen Mode 0



Text screen Mode 1



Text screen Mode 2

Super graphics where they do [illegible] most good

RIGHT from the start I was impressed. As soon [illegible] the initial header of **Galaxia** came up I knew I was in for good graphics – and I wasn't disappointed.

The object of the game, [illegible] of [illegible] set of high quality releases from Kuma, is [illegible] get past the alien fleet and dock with your starship.

"Ah", you're probably thinking. "Just another sideways scrolling screen with lots of obstacles you must dodge". And you'd be right – but you'd be missing the point by about as much as you would if you described the Mona Lisa as a snapshot of a woman.

As you hurtle down the screen you encounter 10 types of alien, each becoming successively more nasty.

If you're the non-violent kind you [illegible] move your craft up and down to avoid them. If you're like me you'll blast them to oblivion with your fire button (my psychiatrist says it's good for [illegible] – mind you, he's got [illegible] Spectrum).

There are Rammers, Mushies, Spiders, Bangers, Bashers, Spores, Darts, Zappers, Swoopers and Baiters – and they all make exceptional use of the Amstrad's graphics.

The [illegible]mmers and Mushies (mushroom shapes) [illegible] easily dealt with, but the Spiders and Bangers (yes, you've guessed – flying sausages) with their erratic movement were harder to negotiate.

Mind you these weren't that difficult. The Bashers [illegible] Spores that come next posed [illegible] first problems. Bashers fire missiles ahead of them – not easy to dodge [illegible] first but I [illegible] got the hang of it. The Spores [illegible] far [illegible] evil – sending their projectiles diagonally [illegible] me. Very nasty.

The darts didn't fire anything – they just hurled themselves towards [illegible] at twice the usual speed.

Most of my early efforts [illegible] [illegible] that stage. The four lives I was given didn't last very long [illegible] when you lose [illegible] you restart at the level of nasty you were up to.

But the game's totally compulsive and I kept at it – only to encounter the super-fast Zappers who [illegible] definitely [illegible] news. Not only are they fast, they fire at you.

After them I took the Swoopers [illegible] my stride – well, almost. These very agile birdies who launch their explosive [illegible] [illegible] you were a little tricky.

Then [illegible] the Baiters – nasty [illegible] devils who hung back and waited for me to [illegible] [illegible] before launching their far too successful attacks.

But, after many many tries, I cleared the lot of them and found my starship – only to crash into the thing when I tried to dock. A space pilot's life [illegible] hell.

But this game is also a great deal of fun. Galaxia's got the lot – good graphics, great sound and high-speed action.

It's [illegible] pity there isn't a joystick option, and that you can't select your own keys. But these minor niggles don't spoil the fun.

I can't stop playing Galaxia, and if I've manged to convey just [illegible] fraction of [illegible] excitement, I'm sure you'll [illegible] why.

Andrew Scriven

A better class of hangman

WHEN I had my Electron, I had **Wordhang** by Bourne Educational Software, and it rapidly became [illegible] great favourite with the younger members of my family.

I was therefore delighted [illegible] see it available for the Amstrad, and doubly delighted because it's even better on this machine. Too often programs are converted from one micro to another without sufficient attention to the strengths or weaknesses of the new one.

Here the sound is much stronger [illegible] the Amstrad version, which [illegible] hardly surprising in view of its vastly superior capabilities.

Hangman has appeared for every machine available, I would imagine, but I've not seen a better version than this. It is crammed with evidence of thoughtful preparation and clear programming. The graphics [illegible] superb, yet not at all frightening – an important point with youngsters.

However my three loved the catchy little fanfare that heralded [illegible] successful result, [illegible] they made [illegible] they didn't hang many!

There is [illegible] extensive built-in vocabulary with six levels, and also the option to enter [illegible] single word at a time for a child to solve.

This is most useful [illegible] the parent can gauge the exact sort of word to stretch the child.

One of the most exciting facilities is the option to make up [illegible] word list oneself which can then be saved to cassette and reloaded at [illegible] later date.

Teachers will also find the monitor facility useful, as [illegible] displays the [illegible] of the last six children to [illegible] the program, their level and the number of successes and failures they had.

Key responses are accurate [illegible] rapid, and [illegible] my children [illegible] anything to go by, [illegible] will outlast many other educational programs [illegible] far as their enthusiasm is concerned.

Phil Tayler

Game has the right spirit . . .

HIDDEN away in the creepy mansion at the top of the hill are a number of power jewels. Many have searched for them but [illegible] one has lived [illegible] tell the tale. Have you the courage to [illegible] and succeed where others have failed before you?

This is the challenge issued by **Ghouls**, Micro Power's first contribution to the Amstrad software market. It is [illegible] translation of an existing game for the BBC Micro and one of their best.

You control the star of the show, a little man looking like pac-man on two legs. His ever munching mouth continuously snaps up titbits for bonus points, [illegible] you attempt to access the various levels in your search for treasure.

You start off in the first of four screens – Spectre's Lair. Here to hinder you in your quest you will find the mansion's ghost who's out to get you.

He's [illegible] comical looking chap with an enormous frown covering most of his face. Let him catch you and you will die, watching that frown change to an enormous cheeky grin. This chap follows you through every screen and at times his presence is positively painful.

You have [illegible] time limit in which to reach the box of

jewels at the top of the screen, in order [illegible] get to the next landing. Titbits and the occasional stray jewel munched en route count for bonus points.

A jewel has the additional perk in that it makes the ghost disappear for a short while. However there are plenty of other hazards to make life unpleasant.

In order to make progress you must balance on [illegible] moving platform and leap to and from it to higher levels. There is also a set of poison-smeared spikes in your way and contact with any one of these will prove fatal.

Should you succeed on the first screen you will progress [illegible] Horrid Hall. Still pursued by the ghost, you will have to cope with not only a moving platform but also contracting floorboards. To get started you have [illegible] your disposal a large spring to propel you upwards to the first landing. There are also more poisonous spikes on this level.

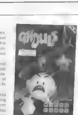
Succeed on screen 2 and Spider's Parlour awaits you, introducing [illegible] obvious adversary. He is something to behold but not to be touched – an additional hazard to be avoided. Fortunately he stays in one spot bouncing up and down waiting patiently for a tasty morsel. Make sure it's not you!

The infuriating aspect of the game, as with most multi-level games, [illegible] that [illegible] soon as you die you start back at the beginning of the screen no matter how far you have progressed.

I must confess, it is because of this that I haven't even [illegible] screen 4 except in the demo mode. (You automatically go into this if you don't start the game within one minute.) To be honest I've only seen screens 2 and 3 by watching the kids play – I can't get to the end of screen 1 myself.

Even so I've seen enough of the game to consider it excellent value. It is extremely addictive as there [illegible] always that incentive [illegible] "crack it this time".

The graphics are well presented but I would have thought better use could have



been made of the colours available on the Amstrad.

An outstanding feature of this package [illegible] the clever use of sound. All kinds of weird and eerie sounds make the game come to life.

My only real criticism [illegible] the [illegible] of preset keys – surely it should be standard today to allow the user [illegible] define his own keys? I think this is [illegible] reason I [illegible] make much progress in these quick reaction games.

There [illegible] a joystick option and [illegible] option to freeze the action while you [illegible] that necessary natural break.

If Micro Power wanted to make [illegible] "spirited" break into the Amstrad market, they certainly picked the right game.

Alan Sergeant

Fancy a flutter?

I CANNOT say that I am an avid football supporter but I do take [illegible] passing interest in the results every Saturday. I have never seriously attempted the pools although I do know what is involved in marking up [illegible] coupon. It [illegible] therefore with some trepidation that I tacked **Amstra-Draw** by B.S. McAlley.

This package [illegible] a pools prediction program which attempts to forecast draws that will [illegible] each week's English and Scottish football league matches.

Once [illegible] program is up and running the database on the [illegible] must be loaded. The opening screen shows [illegible] menu of [illegible] options.

1. Set [illegible] league position records. In order [illegible] function correctly the program must know where in its divisions any team is placed. So each team [illegible] put into one of three sections – top quarter, middle half, or bottom quarter.

The program automatically brings each team's [illegible] to the [illegible] in divisional, alphabetical order. You [illegible] then prompted for its rating. This is the tedious part of the routine and, as the instruction booklet points out, it must [illegible] carried out every week in the early part of the season.

2. Input last Saturday's re[illegible]lts. This must be done every week as the information is used to update the prediction database. The fixture, previously defined with the next option, is displayed with the options to enter the result – 1. Home Win, 2. Away Win, X. Draw, p. postponement.

3. Input next Saturday's matches. These [illegible] the fixtures from which the program will predict the likely draws. The whole division [illegible] displayed, starting with the first, and each team [illegible] numbered.

4. Predict [illegible] Saturday's draws. This, of course, is why you're contemplating buying the software. This option computes the probabilities of each division and asks how many of the most likely draws you require. [illegible] then displays these [illegible] descending order of probability giving the com-

Binary tree is growing

I FOUND **World-Wise** by Bourne Educational Software to be a thoroughly worthwhile and interesting program, which will appeal strongly to parents keen [illegible] use [illegible] machine for educational programs.

Unlike many others, this genuinely is educational because of its open-ended approach. Many teachers and parents may have seen the Tree of Knowledge type of program, in which binary trees are used.

No, binary trees are not the ones you pick silicon chips from! They are a way of teaching the computer more about [illegible] subject, and by doing so to reinforce one's own knowledge.

This program takes the binary tree idea and applies it most successfully [illegible] geography. To use an example from the World version the program can be taught about capital cities (actually there are 10 choices of subject to select from).

[illegible] knows only about two, so if you've not picked the capital of Peru or Sweden, the micro gives up. It then asks for the city you [illegible] thinking of, such as Rome. Next it asks for [illegible] question which will enable it to differentiate between say Stockholm and Rome, and which answer (yes or no) would be correct for London.

It [illegible] this information to its store, and can thus build up [illegible] large databank with the help of the child.

There are two separate programs, one on Britain and one on the World. Each contains data on two entries in each of 10 categories. I imagine it would take a lot of information to fill the memory of the Amstrad.

Sound is used in a rather poor way and I preferred to turn it off, but the screen displays are clear. No nasty colour combinations to strain the eyes here! Thoroughly recommended.

[illegible]hil Taylor

puted chance of the match being a draw.

5. Save Data. Once all your information has been updated, this facility enables you to save the new database.

6. Input individual results. This routine enables you [illegible] input results of mid-week matches without disturbing the match list you created the previous weekend.

7. Perm generator. This calls up [illegible] self contained program which will enable you to complete your pools coupon directly from the monitor.

The screens are well laid out and the [illegible] is easy to use. The small instruction booklet and the on-screen instructions are clear and easy to follow.

I may even take a dip in the pools myself just to give the program a real good testing. The author is quite emphatic that "Wins are not guaranteed". It [illegible] perhaps a little pricey at £9.95 but who knows – this time next week I might be able to afford one!

Alison Blaylock

Why they dig Willy

EVERYONE raved about Miner 2049er for the Atari when it emerged some 18 months ago. I couldn't really see what all the fuss was about.

[illegible] later spawned a glut of multi level type games which flooded the software market and no mean competition was the top selling **Manic Miner** for the Spectrum.

Well the game is now Software Project's first contribution to the Amstrad scene and I must admit I can now see why the game was [illegible] popular. It is fun to play, quite nerve racking, and above all infuriatingly addictive.

Our famous hero Miner Willy, while prospecting down Surbiton, has stumbled on an ancient long forgotten mine shaft. Exploring further he finds evidence of a lost civilisation far superior to our own, where automatons dig deep into the earth's core excavating the raw materials for their industry. And they're still [illegible] it.

Willy realises that vast wealth awaits him if he can only find the underground store. In each of 20 – yes, 20! – chambers, you control the little fellow [illegible] he collects all the flashing keys. Collecting these is your only means of getting [illegible] the next chamber and it's no mean task.

While doing so you must avoid a multitude of horrors – poisonous pansies, spiders, slime, mutant telephones, man-eating toilets, bouncing cheques and many more. A single touch from any one of them [illegible] instant death.

The test is really one of strategy and timing rather than quick reactions although

on occasions it [illegible] help to be a little swift of finger to avoid the nasties.

When you first see the room you must plan [illegible] best, and sometimes the only, route to get all the keys. Once you have mastered a screen you will normally have little difficulty in completing it every time.

There is [illegible] demo at [illegible] start which takes you through each scenario in turn and an impressive colour display it makes.

But I must admit I haven't got past screen 2 yet. I am one of the world's worst games players. I did however watch my lad get to screen 6 with not too much heartache.

There's plenty of choice in the movement keys as the whole of the top row are alternately for left and right.

Very impressive graphics and sound make this a game which must have long lasting appeal to game players of all ages.

David Andrews

Counter attraction

THE Moors Challenge by Timeslip Software is [illegible] version of the popular board game Othello.

For those who have never played Othello I will describe what the game is all about. The playing board is an [illegible] × [illegible] grid onto which two players take [illegible] in turns to place circular discs. These should be positioned in such a way [illegible] to trap the opponent's pieces.

Each player has different coloured counters. The number of counters each person has determines the overall winner when either [illegible] grid has been filled or one player's counters have all been trapped.

When the game has loaded you will [illegible] asked which type of game you would like to play. There [illegible] three options. The first shows [illegible] demo of how the

game is played.

The second option allows you to play against the computer. There [illegible] five levels, from simple to advanced, which can be selected once the board has been [illegible] up.

The third option allows two players to compete. This feature is very useful [illegible] you want to challenge [illegible] friend.

The program is well presented. The instructions [illegible] not too clear, but playing the game helps you understand them more.

The game itself is excellent. Don't be put off if you consider yourself a bit of a wally. The

A question of style

Happy Writing, another from [illegible] Bourne stable aimed [illegible] three to six-year-olds, aims [illegible] demonstrate correct letter formation.

A "magic pencil" with the child's name written [illegible] it draws brightly coloured numbers letters and words showing where [illegible] start the figure, which direction to take and where to end.

The child is expected to copy each figure onto paper [illegible] drawn on the screen. The speed [illegible] which the figure appears [illegible] be adjusted to suit his stage of learning.

You can also tailor the program so that you can practice single letters, both lower and upper case, numbers, a set of similar letters – for example, c, [illegible], d, g – your own special set and [illegible] word or set of words.

The group of three to four-year-olds I worked with had little writing experience and found it difficult to copy the letters [illegible] their paper.

Even with the guidance of the screen they still tended to draw some letters reversed without realising it.

The style of writing the program demonstrates may not appeal to all teachers. For example, the verticals are curled at the bottom.

If however this is the style used in your school it should be useful in giving children practice in writing correctly when used in moderation.

So if you're considering computerising your classroom, Happy Writing may be of use [illegible] you.

But make sure you [illegible] it running first.

Carole Sillers

simple game could be tackled quite easily by an eight-year-old, but the advanced level could cause problems.

Overall, a great game well worth buying.

Susan Tudor

Royal progress

I HATE writing reviews of adventure programs for there [illegible] two insurmountable problems.

Firstly, how much do you tell the players about the game? Do you tell them not to kiss the frog until it's under the rainbow, [illegible] that the friendly dwarf [illegible] a pathological liar? Do you reveal that the pH of the Red River is more than a little on the acid side?

Of course you don't – you'd be spoiling the adventure for the reader, wouldn't you? So rest assured, none of the above refers to **Royal Quest,** from Timeslip.

Even oblique hints like "try reading page 379 of Fraser's The Golden Bough" might offend the adventure purist.

The second problem is worse. You have to get sufficiently far into the adventure to let you feel the tone of it – and, of course, the program[illegible] aren't going to make it easy for you, [illegible] they?

What the timorous [illegible] prays for is a game that starts simply, has enough "classical" themes to give him something to latch onto and then [illegible] or two real problems to let him get the flavour of the game.

Fortunately for me, Royal Quest provides all these. You start on an East-West path, with a palace to the West. I suggest you visit the palace first in order to find out what the game is about (another stolen crown) and to receive your sword.

From then on it's East along the path, with diversions to visit a cemetery and overgrown orchard – picking up one or two things – then via a sleeping dwarf into the forest.

Oh for an adventure without a forest! Still, this one isn't too bad, and you're soon at the edge of the lake where the action really starts.

Incidentally, you need "GO BOAT" here – the only questionable bit of vocabulary I've come across. Normally the commands you enter [illegible] quite sensible – and you can abbreviate them to three letters, with single letters for N, S, [illegible] and W.

All [illegible] all the game is standard, enjoyable adventure fare, with enough tricks and puzzles to maintain interest, plus one or two nice touches. I particularly liked the book packed full of information about wizards, dragons and the like.

And the game was well presented, with text appearing [illegible] a parchment scroll, making [illegible] of windows to separate the room descriptions from the commands and responses.

It was a pity the effect was spoiled by a plethora of misspellings that gave the game a totally undeserved amateurish [illegible] ar [illegible].

That [illegible] apart, Royal Quest is enjoyable [illegible] [illegible]taining. Perhaps not quite a classic, but with enough interest and difficulty to make [illegible] glad of the facility for saving unfinished games.

Andrew Spencer

Borrow a Biggles

WHEN [illegible] [illegible] to flying I'm about as useful as a chocolate fireguard, but I was determined [illegible] try out Anirog's jet airliner simulator **Flight Path 737.**

You take off from an airfield surrounded by high mountains and having climbed safely over them, prepare for a landing [illegible] the valley below.

There are six levels of play ranging from "first solo" to "test pilot". I curbed my natural desire [illegible] try "stunt pilot" and selected the first level.

The [illegible] consists of a split display in Mode [illegible] with the view from the cockpit above [illegible] instrument panel.

The brief pilot's notes explain the take off procedure.

Press 7 and you will start to taxi at [illegible] knots (keys 7,4,1,0 control speed by +20, +10, -10 and -20 knots respectively). I put the flaps down [illegible] requested and tried to line myself [illegible] with the runway.

I found this very tricky [illegible] start with as the key response [illegible] slow and regularly I had overshot the target [illegible] after releasing it (a joystick, [illegible] the left and right cursor keys, control lateral movement). You have to get the centre white line dead straight [illegible] else.

"Increase speed to 180 knots but not [illegible] than 200 because the flaps will be damaged", you [illegible] advised. I must have caused thousands of pounds worth of damage to those flaps before I finally got it right.

Pull the joystick back (or use the down arrow key) and the jet will take off. Sure does! As you climb you have to keep your finger [illegible] the throttle to maintain air speed.

Once above 300 feet you can retract the undercarriage (A) and the flaps (F) – and this is where the trouble starts.

Apparently when you [illegible] the undercarriage on [illegible] aircraft the speed will increase by 5 knots so I had to be careful at this phase not to exceed the maximum flaps-down speed.

We climbed higher and the scenery changed to mountain tops as we levelled out at 6,000 feet. With the mountains approaching a red ground warning flashed but steadied when we were safely above the hazard. When the light went out we were clear of the mountains and ready for landing.

This is a nice piece of software for the flying enthusiast and games player alike and should be very popular.

The graphics [illegible] well done and the sound is adequate. I [illegible] a little dubious about the slow key response but I don't fly enough to know if this is standard on flight simulators. After all you can't turn a 737 round [illegible] a sixpence . . .

If you want a challenge that's going to have you kicking the cat every 5 minutes, then look [illegible] further.

[illegible]len Whitehead

REVIEWED THIS MONTH	
Amstradraw	B.S. McAlley
[illegible] 737	Anirog
Football Manager	[illegible]
Forest at World's End	Interceptor
Galaxia	Kuma
[illegible]	Micro Power
[illegible] Writing	B.E.S.
Manic Miner	Software Projects
Royal Quest	Timeslip
[illegible]	Timeslip
Timeman One	B.E.S.
[illegible]	B.E.S.
[illegible]	B.E.S.
[illegible]	B.E.S.

Be a Big Mal

FOOTBALL Manager by Addictive has [illegible] [illegible] one of the best strategy games available for home computers.

Already a best seller for a number of other machines I have watched my friends become totally enthralled with the game, forgoing meals to have just [illegible] more match.

As the manager of a football club you have to get from league division four to league division [illegible] in as few seasons [illegible] possible.

The first screen asks your name and allows you to select your team from the whole league. There seems to be little advantage in picking the well-known clubs as they are all dependent upon the skill of their manager – you.

The team you inherit is then displayed, together with six pieces of information on each player – playing position, number, skill rating, energy rating, saleable value and

status (either picked or injured).

From this screen there are a number of other options affecting the game. You can get a display of your performance, beg for [illegible] loan from the bank and even pay off loans.

You can also affect the game settings if you wish. You can change your skill level, change the team and player names, [illegible] [illegible] game or restart a saved game.

Once you have done all the housekeeping from the first menu you have an announcement of your next fixture, which might be in either League or Cup competition.

Following this there [illegible] a readout of the comparison of the teams giving ratings for Energy, Morale, Defence, Midfield and Attack. This is worth careful study as it is here that you can achieve the most valuable changes by modifying your team slightly.

Presuming that you have enough players on your books you can arrange [illegible] strengthen your team to meet the opposition's rating. It [illegible] said that one of the most important skills of management is the resting of players who then build up their strength and morale. It certainly seems to matter in this game.

You have chances to [illegible] your [illegible] players and [illegible] for others. Unsuccessful bids [illegible] an increase in the asking price so [illegible] warned.

I doubt that this game will [illegible] be bettered. [illegible] is all it claims to be and is very addictive [illegible] boot.

Dave Carlos

Time...

TIMEMAN One, from Bourne Educational Software, is designed to help children aged 4 to 9 to tell the time. The first stage of the program, for the younger child, introduces the hour hand only, the second stage the minute hand, and the third and final stage combines the two.

Having taught how to [illegible] the time, the program then provides three similar stages to enable children [illegible] set the clock. Time is expressed throughout in minutes past the hour in line with digital clocks and watches.

The graphics are appealing. There is [illegible] large clock face, and [illegible] man placed half way up [illegible] ladder. The program asks: "What time [illegible] it?" If the child [illegible] correctly by pressing the appropriate number of keys the man climbs two steps up the ladder.

When the [illegible] is wrong he goes down a step, the child is told what [illegible] wrong and given another [illegible]. When the man reaches the top he dances a [illegible] and plants a flag.

Setting the clock [illegible] again [illegible] three stages – first setting the hour hand only by pressing the H key to rotate the hand, and then the minute hand by pressing the [illegible] key. This can [illegible] done in [illegible] [illegible] five minute intervals [illegible] required. Finally both hands [illegible] be adjusted together.

This is [illegible] very handy package which the children will enjoy working through.

It [illegible] [illegible] wide range of abilities – suitable for [illegible] to 5-year-olds who can practice telling the time by the hour hand only and 7 to 9-year-olds who can use [illegible] hour and minute hands. Definitely worth having in the classroom.

Carole Sillers

...and time again

TIMEMAN Two follows on from Timeman One but can equally well be used on an independent basis. It is designed to give children aged 4 to 10 [illegible] greater understanding of time, including the 24 hour clock.

[illegible] covers minutes to the hour, the quarter and half hours as well as the 24 hour clock. As [illegible] Timeman One, the children are required to tell the time and to [illegible] the clock. This can be in one or five minute intervals to suit each child.

As before, a little man climbs up [illegible] ladder each time the child gives the correct [illegible].

Again a very useful program giving children the practice they need in telling the time, but [illegible] a way that is not boring.

Carole Sillers

Superb scenery at World's End

AS [illegible] as I saw the opening screen of **Forest [illegible] the World's End,** the latest adventure from Interceptor Software, I knew I [illegible] in for a real treat.

To say that the title page shows a castle on top of a hill, surrounded by trees is accurate, but [illegible] totally fails to convey the quality of graphics involved.

Quite simply they're superb – better than anything I've seen on other micros.

As the tape finishes loading you find yourself in the Great Valley before a forest – again beautifully drawn, conveying that ethereal air so vital for an adventure.

You won't get [illegible] picture for every location you visit, but still there are [illegible] enormous number packed into the game. I could continue raving about the graphics, but, after all, it's supposed to be [illegible] adventure game not a picture book, so how good is it?

Well, it's not bad [illegible] all. You'll find plenty to intrigue and bewilder you as, in your role as the mightiest of ancient warriors, you try to [illegible] Princess Mara from the clutches of Zarn, [illegible] evil wizard.

However I get the impression that more effort has gone into the graphics than the plot, which is rather predictable.

All that is, except the combat. Having collected your bow and [illegible] you are ready [illegible] repulse attackers (hint – go south first).

But what a boring business it is. No skill, just totally random results [illegible] you loose your arrows. I find this aspect of adventures a definite minus, though I admit that I'm the type who prefers chess to Monopoly.

In mitigation, there is the Save feature, careful use of which can make the random mortality less final.

By the way, to fire your arrows enter: KILL ELF WITH BOW. It's the bow you use, not the arrows!

A [illegible] feature of the game [illegible] the command syntax. As shown above, you can get away with far more than the normal two word commands of other adventures.

The game is thoroughly enjoyable, even if the objects do tend to get presented to you on [illegible] plate.

The first major obstacle is the chasm, which isn't a problem provided you've thoroughly looted all the other locations on this side of it.

Then there's a journey over the plain to the bank of a river – don't cross, though, until you've paid the old well a visit.

After one or two more incidents you come to a precipice. No problem provided you remember there's [illegible] dragon watching you.

Then you'll want to pay the witch a visit before you go through the Marl's gateway . . .

As you [illegible] [illegible], there's plenty of action, and your interest level is maintained throughout. All right, there may not be much innovation, but that's not Mills and Boon's strong point either, and they still sell.

And, of course, the graphics take it well out of the ordinary class of adventure games – although I would have liked them to be an integral part of the plot rather than merely decorative.

Still, Forest at the World's End [illegible] an extremely enjoyable, entertaining game and deserves to be successful.

After this good start, I look forward to more graphic adventures from Interceptor. They're going to be even better.

Justin Marriot

Meet Smiley and beat the galloping Grumpies in ROLAND WADDILOVE's version of [redacted] old favourite

Introducing Smiley...

I [redacted] this colourful and compulsive game you have to guide Smiley round [redacted] maze picking up coloured buttons that are scattered about. The buttons belong to the Grumpies, who take exception to you pinching them from under their nose and chase you round the maze, trying to catch you.

If you collect all the buttons in the maze it [redacted] redrawn and the game starts all over again, except that you and the Grumpies move a little faster.

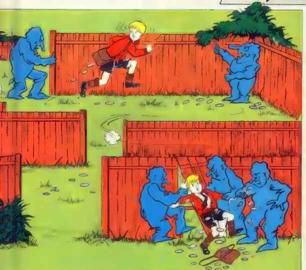
Yes, it's an Amstrad version of the old favourite – no collection is complete without it! There is a bonus which decreases with time, three lives and a keyboard or joystick option.

The program is fully structured with no GOTOs to confuse you. It is controlled by calling the various subroutines from lines 40 to 210 to print the instructions, draw the maze and move the characters.

Each subroutine has been given a title in [redacted] REM statement at the start of it so you know where to look if it does not work first time (it's nearly impossible to type in a program without making at least one mistake).

SUBROUTINES

Timer interrupt	I haven't worked out how to [redacted] the clock yet [redacted] I have used Timer 1 to generate an interrupt and [redacted] my [redacted] clock.
Bonus interrupt	Timer [redacted] is the bonus counter, counting down to zero in 100 seconds. The bonus is printed in window 1 so as not to upset the print positions and colours of the other characters.
Initialise	Dims the arrays, sets the colours and the high [redacted]. k% is the "Y" key.
Draw the [redacted]	Reads the data statements and [redacted] to the Ascii code of each character before printing.
Start positions	Sets the start positions of the Grumpies and Smiley. Sets the bonus and the dots left. Reduces delays.
Set up	Sets score and delays to initial values. Prints score and three Smileys.
Move man	If it is not time then return. Sets the next time. Finds the new coordinates, and sets the flag (ok) if they are the same [redacted] of the Grumpies. Prints at the new position, [redacted] the old [redacted]. Increments score [redacted] button has been picked up.
Move ghosts	Calls ghost to move the Grumpies if it is time.
Ghost	Looks where Smiley [redacted] and moves towards him if [redacted] wall [redacted] not in the way. Checks if he has been caught. Replaces button if necessary.

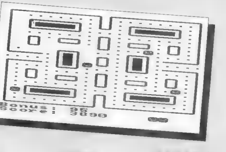


Caught	Makes a rude noise and flashes several characters. Decreases lives.
Instructions	The large title is drawn by printing it at the bottom in black, looking at the points and plotting it at the top of the screen. Prints the instructions, sets either keyboard (default setting) or joystick mode.
Next	Prints screen completed in transparent mode. Adds the bonus to the score and increments the screen.
Game	Clears the screen by drawing black lines towards the centre. Prints scores. Asks if you want to play again.

VARIABLES

t%	Timer.
bonus%	Bonus.
maze$(30)	The maze.
ghost%(2,1)	Grumpies' positions.
gtime%(2)	Time when the Grumpies next have to move.
gdelay%(2)	Delay for the Grumpies.
i%,j%	Used in loops.
hi%	High score.
k%	General variable.
ok	A flag to show whether caught.
manx%,many%	Smiley's coordinates.
x%,y%	Temporary coordinates of Grumpies or Smiley.
score%	Score.
dots%	How many dots left.
lives	Number of lives left.
screen	Number of screen.
a%,b%,c%,d%	Values used by INKEY().
mtime%	The time Smiley can next move.
mdelay%	The delay for Smiley.

```
10 [redacted] **** SMILEY ****
[redacted] REM *By R.A.Waddilove*
30 :
[redacted] MODE 1:GOSUB 1400
50 MODE 0:GOSUB [redacted]
60 WHILE k%=43
70 GOSUB 980
[redacted] lives
[redacted] 400:GOSUB 810
[redacted] EVERY 2,1 GOSUB 240
110 EVERY 50 GOSUB [redacted]
120 WHILE [redacted] dots%
130 [redacted]
[redacted] WEND
150 [redacted]
160 IF [redacted] 1660 ELSE [redacted]
1310
170 [redacted]
180 [redacted] 1720
190 [redacted]
[redacted] 1
210 WHILE INKEY$<>"":WEND
220 END
230 :
[redacted] t%=t%+1:RETURN :REM timer interup
t
250 :
[redacted] bonus interupt **
270 IF bonus% [redacted] bonus%=bonus%-1:LO
CATE #1,7,24:PRINT #1,bonus%:RETURN E
LSE RETURN
280 :
290 REM ** Initialise **
[redacted] DIM maze$(30),ghost%(2,1),gtime%(
2),gdelay%(2)
310 [redacted]
[redacted] i%=0 TO 15
330 READ j%:INK i%,j%
[redacted]
[redacted] 0
[redacted] k%=43:hi%=0
[redacted]
380 :
390 REM [redacted] Draw the maze **
[redacted] LOCATE 1,1
410 RESTORE [redacted]
[redacted] FOR i%=1 TO 23
430 READ maze$(i%)
[redacted] j%=1 TO 20
[redacted] k%=82+ASC(MID$(maze$(i%),j%,1))
[redacted] IF k%<147 THEN PEN screen ELSE PE
N screen+1
470 PRINT CHR$(k%);
[redacted]0 NEXT
[redacted]0 NEXT
[redacted] RETURN
[redacted] :
520 DATA DHHHHHHHHJDHHHHHHHHJ
530 DATA C>>>>>>>>CC>>>>>>>>C
[redacted] DATA C>DHHHHJ>CC>DHHHHJ>C
[redacted] DATA C>C====C>AB>C====C>C
560 DATA [redacted]
[redacted] DATA C>>>>>>>>DJ>>>>>>>>C
[redacted] C>DJ>DHJ>CC>DHJ>DJ>C
590 DATA [redacted]
[redacted] DATA C>AB>>>>>AB>>>>>AB>C
610 DATA C>>>>DHJ>>>>DHJ>>>>C
[redacted] DATA AHHH>C=C>DJ>C=C>HHHB
[redacted] DATA >>>>>C=C>CC>C=C>>>>>
640 DATA DHHH>C=C>AB>C=C>HHHJ
650 DATA C>>>>AHB>>>>AHB>>>>C
660 DATA C>DJ>>>>>DJ>>>>>DJ>C
670 DATA C>CC>DHJ>CC>DHJ>CC>C
680 DATA C>AB>AHB>CC>AHB>AB>C
[redacted] DATA C>>>>>>>>AB>>>>>>>>C
[redacted] DATA C>DHHHHJ>>>[redacted]>C
[redacted] DATA C>C====C>DJ>C====C>C
[redacted] C>AHHHHB>CC>AHHHHB>C
730 [redacted] C>>>>>>>>CC>>>>>>>>C
740 DATA AHHHHHHHHBAHHHHHHHHB
750 :
760 DATA 0,1,7,5,12,3,9,4,10,0,12,24,
6,11,26,18
770 :
[redacted] 19,2, 19,22, 2,22
790 :
[redacted] REM ** Start positions **
[redacted] RESTORE [redacted]
[redacted] mdelay%=mdelay%-1:mtime%=mdelay%
[redacted] manx%=2:many%=2:ok=-1:t%=0
[redacted] 14:LOCATE 2,2:PRINT CHR$(224)
[redacted] i%=0 [redacted] 2
[redacted] gdelay%(i%)=gdelay%(i%)-2*(i%+1):
gtime%(i%)=gdelay%(i%)
[redacted] x%,y%
[redacted] ghost%(i%,0)=x%:ghost%(i%,1)=y%
[redacted] 11+i%:LOCATE x%,y%:PRINT CHR$
(225)
[redacted] NEXT
[redacted] maze$(2)=LEFT$(maze$(2),1)+" "+MI
D$(maze$(2),3)
[redacted] score%=score%+10:dots%=183
[redacted] PEN 14:LOCATE 15+lives,25:PRINT "
 ";
[redacted] bonus%=100:PEN #1,10:LOCATE #1,1,
24:PRINT #1,"Bonus:";bonus%
950 RETURN
960 :
[redacted] ** [redacted] up **
[redacted] CLS:PEN 15:LOCATE 1,25:PRINT "Sco
re:";0;
990 PEN 14:LOCATE 16,25:PRINT CHR$(22
4);CHR$(224);CHR$(224);
[redacted] gdelay%(0)=20:gdelay%(1)=40:gdel
ay%(2)=80
1010 mdelay%=6:score%=0:bonus%=0:scre
en=1:lives=3
1020 RETURN
1030 :
[redacted] **
[redacted] IF mtime%>t% THEN [redacted]
[redacted] mtime%=t%+mdelay%:x%=manx%-(INKE
Y(d%)>-1)+(INKEY(c%)>-1):y%=many%-(IN
KEY(b%)>-1)+(INKEY(a%)>-1)
[redacted] IF x%=0 [redacted] x%=20
1080 IF x%=21 THEN x%=1
[redacted] IF (x%=ghost%(0,0) [redacted] y%=ghost%
(0,1)) OR (x%=ghost%(1,0) AND y%=ghos
t%(1,1)) [redacted] (x%=ghost%(2,0) [redacted] y%=gh
ost%(2,1)) [redacted] ok=0
[redacted] MID$(maze$(y%),x%,1)">" [redacted]
 RETURN
1110 PEN 14:LOCATE manx%,many%:PRINT"
 ":LOCATE x%,y%:PRINT CHR$(224):manx%
=x%:many%=y%:IF MID$(maze$(y%),x%,1)=
" " THEN [redacted]
[redacted] 129,30,5,15:dots%=dots%-1:
score%=score%+10:PEN 15:LOCATE 7,25:P
RINT score%;:maze$(y%)=LEFT$(maze$(y%
),x%-1)+" "+MID$(maze$(y%),x%+1)
1130 [redacted]
1140 :
1150 REM ** [redacted] ghosts **
1160 FOR i%= 0 TO 2
1170 IF gtime%(i%)<t% THEN [redacted]
[redacted]
1190 [redacted]
1200 :
1210 [redacted] ** ghost [redacted]
1220 SOUND 130,370*i%+370,10,14:gtime
%(i%)=t%+gdelay%(i%)
[redacted] y%=ghost%(i%,1)+(ghost%(i%,1)>ma
ny%)-(ghost%(i%,1)<many%):IF MID$(maz
[redacted](y%),ghost%(i%,0),1)>"?" THEN y%=gh
ost%(i%,1)
[redacted] x%=ghost%(i%,0)-(ghost%(i%,0)<ma
nx%)+(ghost%(i%,0)>manx%):IF MID$(maz
e$(y%),x%,1)>"?" THEN x%=ghost%(i%,0)
[redacted] IF [redacted](maze$(ghost%(i%,1)),ghos
t%(i%,0),1)=">" [redacted] temp$=CHR$(144)
[redacted] temp$=" "
[redacted] PEN screen:LOCATE ghost%(i%,0),g
host%(i%,1):PRINT temp$:PEN 11+i%:LOC
ATE x%,y%:PRINT CHR$(225):ghost%(i%,0
)=x%:ghost%(i%,1)=y%
[redacted] IF x%=manx% [redacted] y%=many% [redacted] ok
[redacted]
1280 [redacted]
```




```
1290 :
[redacted] ** Caught [redacted]
[redacted] i%=255 [redacted] -1
1320 [redacted] INT(RND*15):LOCATE manx%,many%:PRINT CHR$(i%)
[redacted] CALL &BD19:BOUND 132,INT(RND*1000)+1000,5,15
1340 [redacted]
[redacted] lives=lives-1
[redacted] IF lives=0 [redacted] LOCATE 6,12:PEN 14:PRINT CHR$(22);CHR$(1);"GAME [redacted] ";CHR$(22);CHR$(0)
[redacted]
[redacted] :
[redacted] Instructions **
[redacted] 1,10,1,100,10,-1,20
1410 BORDER 0:INK 0,0:INK 1,0:INK 2,6:INK 3,24
1420 [redacted] 1:LOCATE 1,25:PRINT "Smiley v The Grumpies";
[redacted] 1,60,1100,4,1:SOUND 2,80,1100,2,1:SOUND 4,95,1100,8,1
[redacted] TO [redacted] 2
[redacted] FOR y%=0 [redacted]
1460 IF TEST(x%,y%) THEN PLOT x%+120,368+y%*2,3:PLOT x%+120,368+y%*2,3
[redacted] NEXT
[redacted] NEXT
[redacted] LOCATE 1,25:PRINT SPACE$(25);
[redacted] 1,11:PEN 1:LOCATE 1,7
1510 [redacted] "Guide Smiley around the [redacted] collecting":PRINT:PRINT"the coloured buttons, but watch out for"
[redacted] PRINT:PRINT "the grumpies who try to catch you..."
1530 PEN 3:PRINT:PRINT:PRINT
[redacted] PRINT "Press :-"
1550 PEN 1:PRINT:PRINT " K  for keyboard."
[redacted] PRINT:PRINT " J  for joystick."
1570 PEN 2:LOCATE 8,25:PRINT "( Press space to start )"
[redacted] a%=69:b%=71:c%=39:d%=31
1590 WHILE INKEY(47)=-1
1600 [redacted] INKEY(37)>-1 THEN LOCATE 1,17:PRINT CHR$(243):PRINT:PRINT " ":PEN 3:LOCATE 1,22:PRINT "A...up  [redacted]...down  (...left  )...right":PEN 2:a%=69:b%=71:c%=39:d%=31
[redacted] IF INKEY(45)>-1 [redacted] :LOCATE 1,22:PRINT SPACE$(40):LOCATE 1,17:PRINT " ":PRINT:PRINT CHR$(243):a%=72:b%=73:c%=74:d%=75
[redacted] WEND
[redacted]
[redacted] RETURN
1650 :
[redacted] REM Next [redacted]
1670 [redacted] 2,12:PRINT [redacted];CHR$(1);"Screen";screen;"completed";CHR$(22);CHR$(0)
[redacted] screen=screen+1:IF screen=10 [redacted] screen=1
[redacted] score%=score%+bonus:PEN 15:LOCATE 7,25:PRINT score%;
1700 RETURN
1710 :
[redacted] REM ** [redacted]
1730 FOR t%=0 TO 500[redacted]
1740 [redacted] x%=0 TO [redacted] STEP [redacted]
1750 MOVE x%,0:DRAW x%,400,0
1760 MOVE 640-x%,0:DRAW 640-x%,400,0
1770 NEXT
1780 IF score%>hi% THEN hi%=score%
1790 PEN 14:LOCATE 1,3:PRINT STRING$(20,"*")
[redacted] PEN 11:PRINT "Best score=";hi%
1810 PRINT:PRINT:PEN 12:PRINT "Your score=";score%
[redacted] PRINT:PEN 14::PRINT STRING$(20,"*")
1830 [redacted] 13:LOCATE 1,15:PRINT "Another [redacted]...?":k%=-1
1840 [redacted] k%=-1
1850 IF INKEY(43)>-1 [redacted] k%=43
[redacted] INKEY(46)>-1 [redacted]
1870 [redacted]
[redacted] RETURN
```

Give your fingers a rest . . .

All the listings from this month's issue [redacted] available on cassette. See our special offer on Page 55.

Let's lift the hex on hexadecimal

WELCOME to Bits and Bytes, the first in a series of articles in which [illegible] hope to take the mystery out of understanding the fundamentals of the Amstrad's workings.

All too often even competent Basic programmers tend to shy off such topics as binary coding, hexadecimal and assembly language because it seems too "mathematical".

This is a great pity, because the CPC464 is so constructed that a little knowledge in these fields allows you to take full advantage of its advanced facilities.

The mathematical aspects of the subject aren't [illegible] all deep – certainly anyone who can follow Basic should be able to cope with this series.

If you feel that despite our best efforts [illegible] still haven't explained something fully enough, please write in and tell us – we'll try to rectify the situation in later articles.

First we are going [illegible] look at binary code – a way of handling numbers essential to our understanding of what goes [illegible] inside a computer.

Binary is just [illegible] way of coding numbers in a way particularly suitable for computers. It's actually quite simple. What often confuses begin[illegible] is the fact that the binary system codes numbers in a way that can look extremely like the way [illegible] normally code numbers.

For example, if you were presented with a number 100, you would probably decode it in your normal way and say it was "one hundred".

That, however, is just one way of interpreting it. If you decided to decode it as a binary number, you would interpret 100 in a completely different way and say it meant the number "four." *(Never mind exactly how you arrived at that conclusion for the moment.)*

This is what often causes problems – people are so used to dealing with their numbers in the normal way that 100 [illegible] always "one hundred" to them, and they can't make the shift necessary to decode it in binary [illegible] "four".

Actually it is rather ambiguous. Presented with 100, do you interpret it as "one hundred" or "four"? Our rule will be, if you [illegible] our usual way of dealing with numbers *(the hundreds, tens and units you learnt at school – [illegible] put it more formally, the denary system)* you write the number in the normal way.

If you wish the number to be decoded as a binary number you put the symbol '%' in front of [illegible] – 100 means "[illegible] hundred" while %100 means "four".

So far so good. We [illegible] have [illegible] marker (%) [illegible] [illegible] [illegible] that we have to

By MIKE BIBBY

decode the number in [illegible] special way as [illegible] binary number.

However, before you decode you need a rule for decoding – [illegible] how do you get the number "four" from %100? What's the rule?

Let's take a detour for the moment, and think about the coins [illegible] use every day. Our currency consists of these coins:

50p, 20p, 10p, 5p, 2p and **1p** (ignoring the half-pence). We can combine them to give any sum [illegible] wish. For example:

75p is **50p + 20p + 5p**
or **50p + 10p + 10p + 5p**

and [illegible] [illegible]. We are all familiar with this – often [illegible] use multiples of coins to make up a sum. For example, 5p [illegible] be **2p + 2p + 1p**

Using the same coin twice, though, often means that we end up carrying unnecessary amounts of change, and I for one don't like doing that.

Sometimes, however, with our present coinage system we have to use the [illegible] coin twice to obtain certain sums. You cannot, for instance, make up the sum of 4p without doubling up on coins. To avoid repeating coins we would have to invent a 4p coin!

Let's do that; in fact, let's invent a coinage system where you never have to use the same [illegible] twice.

First of all we would need a **1p** coin and, of course, [illegible] **2p** coin, because [illegible] cannot use **1p + 1p** for 2p – it breaks the rule!

Now 3p can be made up of **1p + 2p,** but for 4p we'll have to invent a **4p** coin.

Equipped with that we can make 5p (**4p + 1p**), 6p (**4p + 2p**), and 7p (**4p + 2p + 1p**). In obtaining 7p we used all our available coins, so now [illegible] have to invent [illegible] **8p** coin. If you work it out (and I suggest you have a go) you will find that with the coins you have at your disposal (**8p, 4p, 2p, 1p**) you can make any sum up to 15p. Then you would have to invent a new coin, **16p.**

Notice how the coins we have created have doubled in value: **1p, 2p, 4p, 8p, 16p.** No prizes for guessing what the next one is.

Let's summarise our results in a table (Figure I). Here I have used the columns to show the coins available and the rows to show how the various

totals are made up. A 1 in a particular column means that we use that column's coin, and 0 means that we don't use it. Look at the row for 5p. It has 101 on it. According to our rule, this means we pick out the coins **4p** and **1p** (and NOT **2p**) to make up the 5p total.

	4p	**2p**	**1p**	
%	**1**	**0**	**1**	
→	**4p**	**+**	**1p**	**= 5p**

Now let's get back to computers by dropping all this talk about coins and redraw Figure I [illegible] show the same information but without referring to money – just numbers. Figure II is the new table.

As you can see, there is little change, and we can use this table to encode numbers in general, not just coins. We call this method of encoding the binary system.

Remember, to show that we [illegible] a binary number we precede it with %. So if you see, for example, %101 means:

	4	**2**	**1**	
%	**1**	**0**	**1**	
→	**4**	**+**	**1**	*= 5*

that is we add together the values of the columns containing [illegible]. Look at row 5 of the table to check it. Similarly, %101 would [illegible] 13 in the denary system since

	8	**4**	**2**	**1**	
%	**1**	**1**	**0**	**1**	
→	**8**	**+ 4**	**+**	**1**	*= 13*

By now you should be able to work out for yourself why %100 represents four. From the table, or by using the addition method I've just illustrated, see if you can decode the denary values of the following binary numbers:

%1001
[illegible] 101
[illegible] 11
%1101
% 111

You can use the program accompanying this article to check your results. You've probably noticed by now that [illegible] the binary system you only [illegible] two symbols, 0 and 1, to encode numbers – hence binary. [illegible]i- for two [illegible] in bicycle.

You can encode any number that you want in binary – just [illegible] more columns (or "bits" as we say in computer jargon), remembering that each new bit is worth double the preceding bit.

However it does get terribly cumbersome. For example, 100 (denary) encoded in binary is %1100100 since:

	64	**32**	**16**	**8**	**4**	**[illegible]**	**1**
%	**1**	**1**	**0**	**0**	**1**	**0**	**0**

→ 64+32+4=*100*

It is much easier to handle the number [illegible] our normal system. To a computer this presents no problem, and the fact that binary only uses two symbols is [illegible] bonus because you can represent numbers with a sequence of "switches".

Switches are what [illegible] call "two state" – they're either ON or OFF. If [illegible] have [illegible] sequence of four switches together we can encode numbers by having them either ON or OFF. We could use ON to mean a 1, and OFF to mean [illegible] 0 in a particular column:

	8	**4**	**[illegible]**	**1**	
	ON	**OFF**	**ON**	**[illegible]**	
→%	**1**	**0**	**[illegible]**	**1**	*= 11*

Each of these "switches" represents a bit, and a computer memory is full of bits. The Z80, which is the microprocessor at the heart of the

	COINS			
TOTALS	8p	4p	2p	1p
1p				1
2p			1	0
3p			1	1
4p		1	0	0
5p		1	0	1
6p		1	1	0
7p		1	1	1
8p	1	0	0	0
9p	1	0	0	1
10p	1	0	1	0
11p	1	0	1	1
12p	1	1	0	0
13p	1	1	0	1
14p	1	1	1	0
15p	1	1	1	1

Figure I

Denary Value	Column or Bit Values				Binary Value
	8	4	2	1	
1				1	%1
2			1	0	%10
3			1	1	%11
4		1	0	0	%100
5		1	0	1	%101
6		1	1	0	%110
7		1	1	1	%111
8	1	0	0	0	%1000
9	1	0	0	1	%1001
10	1	0	1	0	%1010
11	1	0	1	1	%1011
12	1	1	0	0	%1100
13	1	1	0	1	%1101
14	1	1	1	0	%1110
15	1	1	1	1	%1111

Figure II

Amstrad system, deals with 786,432 of them.

To make things simpler the Z80 handles the bits in groups of eight bits at a time – the group of eight being called a byte.

With this type of organisation the largest number you can store in [illegible] byte is 255 since:

128 64 32 16 8 4 2 1
% 1 1 1 1 1 1 1 1
→ 128+64+32+16+8+ 4+2 +1=*255*

Of course the computer can handle larger numbers *(and not just whole numbers)* but to do so it must use more than one byte.

Converting a byte from binary to denary is fairly straightforward. Simply write it down under the appropriate column (or bit) values and add together the value of all the columns in which a 1 occurs. For example, given %10010101 you translate [illegible] follows:

128 64 32 16 8 4 2 1
% 1 0 0 1 0 1 0 1
→ 128+16+4+1=*149*

Going from denary to binary [illegible] not at [illegible] difficult, but [illegible] rather hard to put into words. You do [illegible] by subtracting from the number you want to encode the value of each column [illegible] turn, starting with the highest (i.e. 128, 64, 32 and [illegible] on).

[illegible] you can subtract [illegible] particular column value you put [illegible] 1 in that column and continue [illegible] subtract the next lower column value from the remainder.

[illegible] you cannot manage the subtraction you put a 0 in that column and try to repeat the subtraction with the next lower column number.

So, starting with the highest column number (128 in our case), you:

WHILE you have done all eight columns

1. Attempt to subtract the relevant column number (highest first).
[illegible]. IF you succeed then put a 1 in that column number and continue to subtract other columns from the remainder. ELSE put [illegible] 0 in that column.

WEND

Figure III should make it clearer.

In practice, when faced with encoding a number from denary to binary I tend to do it [illegible] my head, seeing which column values will add together to make the sum required, starting with the highest first.

For example, if I [illegible] to encode 161 in binary I would say, "Well, I can [illegible] 128, [illegible] that leaves [illegible] 33 to find. 33 can be made up of 32 and 1 so that does it: 128+32+1=161. So I encode it as:

128 [illegible] 32 16 8 4 2 1
% 1 0 1 0 0 0 0 1
= %10100001

After a while you'll find this way quite simple.

To finish off, I'll leave you with a program to print out the binary value of a number between 0 and 255 (i.e. that can be stored in one byte). Try it with various values and [illegible] if you can accept the results.

The program itself uses one or two ideas, such as AND, that may not be too familiar to you as yet.

```
10 ■■ ************************
■■ REM *** ■■■■■ CPC464 ***
■■ REM ************************
40 MODE 1
50 WHILE -1
60 number = -1
70 WHILE (number > 255 OR number < 0)
■■  LOCATE 1,1: PRINT STRING$(24," ")
: LOCATE 1,1
90 INPUT "Number";number
■■ WEND
110 PRINT:PRINT"In binary ";number;"
is: ";
■■ PRINT BIN$( number,8)
■■ LOCATE 1,9: PRINT"This works out
as"
140 FOR i = 7 TO 0 STEP -1
150 LOCATE 38-4*i,7: ■■■■ USING "###
";2^i
160 LOCATE 38-4*i,9: PRINT USING "###
";( 2^i ■■ number) / 2^i
170 PRINT:PRINT
180 NEXT
■■ ■■■
```

Worry not, "Bits and Bytes" will cover them. Watch this space . . .

		128	64	32	16	8	4	2	1
149									
-128	128 goes – set to 1	1							
21	64,32 can't go – set to [illegible]		0	0					
-16	16 goes – set it to 1				1				
5	8 can't go – set to 0					0			
-4	4 goes – set to 1						1		
1	2 can't go – set to 0							0	
-1	1 goes – set to 1								1
0		% 1	0	0	1	0	1	0	1

Figure III

Learn to use the keyboard

By PHIL TAYLER

MANY [redacted] owners of the CPC464 must already be experienced users, having advanced [redacted] I have through other micros. But for the total beginner the idea of typing in listings from magazines must seem daunting.

Your typing may lack the necessary speed and, in particular, the accuracy that [redacted] so vital. So here is Letter-Litter – a simple but enjoyable way of getting to know the keyboard of your Amstrad micro.

After selecting how long each round should last [redacted] series of letters [redacted] other symbols appear individually on the screen. A little man, already provided by the Amstrad as CHR$(248) will move automatically in the correct direction, provided that the right keyboard response has been made.

A simple hi-score table [redacted] included and the sound is modest, to say the least.

Now you'll need to find something to do with your new-found keyboard skills!

VARIABLES

x% and y%	Coordinates of the man as he moves around the screen.
rx% and ry%	Coordinates of the letter placed on the screen.
char%	Random value which gives [redacted] random letter/symbol of char$
score% and hiscore%	Need no explaining!
name$	Current hi-scorer

Notes

The main game loop is enclosed [redacted] a WHILE WEND loop between lines 130 and 290. The subroutine at lines 720 to 1040 sets up the title screen, etc. The subroutine at lines 350 to 440 selects and places a random letter on screen. The one at lines 450 to 710 is brought into play at the end of the time limit.

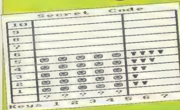
```
10 REM********* letter litter  *****
20 REM************** by **************
30 REM********* phil tayler *********
40 REM*******************************
[redacted] 720 : REM set up titles etc.
[redacted] hiscore%=0
[redacted] x%=1:y%=1:score%=0
[redacted] t=TIME
90 RESTORE
100 gametime=(TIME-t)/300
110 [redacted] 340 : REM to select letter
etc.
120 REM main [redacted] loop begins here
130 WHILE x%<>rx% [redacted] y%<>ry%
140 a$=INKEY$:IF (a$<>char$ [redacted] a$<>"
") THEN SOUND 1,500,50,7:WHILE SQ(1)>
=127:WEND
150 IF a$<>char$ THEN 140
160 SOUND 1,100,3:SOUND 1,100,2,0
170 WHILE SQ(1)>=127:WEND
180 [redacted] x%,y%
[redacted] IF x%<rx% THEN [redacted]
[redacted] IF x%>rx% [redacted]
[redacted] IF y%<ry% [redacted]
[redacted] IF y%>ry% THEN [redacted]
[redacted] PRINT CHR$(7)
[redacted] GOTO [redacted] : [redacted] if it makes this l
ine, then x%=rx% [redacted] y%=ry%
250 PEN 2:PRINT CHR$(32):x%=x%+1:PEN
3:LOCATE x%,y%:PRINT CHR$(248):GOTO 1
[redacted]
260 PEN 2:PRINT CHR$(32):x%=x%-1:PEN
3:LOCATE x%,y%:PRINT CHR$(248):GOTO 1
[redacted]
270 PEN 2:PRINT CHR$(32):y%=y%+1:PEN
[redacted] x%,y%:PRINT CHR$(248):GOTO 1
[redacted]
[redacted] PEN 2:PRINT CHR$(32):y%=y%-1:PEN
3:LOCATE x%,y%:PRINT CHR$(248):GOTO 1
[redacted]
[redacted] WEND
[redacted] score%=score%+1
[redacted] IF gametime>timeallowed [redacted] 450
[redacted] LOCATE x%,y%:PEN 2:PRINT CHR$(32)
:GOTO 100
[redacted]
340 REM ****************************
[redacted] ********* initialization ****
360 [redacted] ****************************
370 [redacted] 0:[redacted] 0,20:BORDER 2:PAPER 2:
[redacted]
[redacted] rx%=(RND(1))*20:ry%=(RND(1))*25:I
F rx%>19 [redacted] rx%<1 OR ry%<1 OR ry%>24
THEN 380
390 char%=(RND(1))*90+33:char$=CHR$(c
har%)
[redacted] IF char%<97 [redacted] char%>90 THEN [redacted]
410 LOCATE rx%,ry%:PAPER 1:PEN 3:PRIN
T char$
[redacted] 2
[redacted] WHILE INKEY$<>"":WEND
440 RETURN
450 [redacted] ****************************
[redacted] REM [redacted] of game procedures *
[redacted] REM ****************************
[redacted] IF score%<hiscore% THEN 640
490 CLS
[redacted] FOR note=1 TO 5:READ a,b,c,d:SOUN
[redacted] a,b,c,d:NEXT note
[redacted] DATA 1,478,50,12,1,426,50,12,1,37
9,50,12,1,358,100,12,1,478,100,12
[redacted] 3
530 LOCATE 5,5:PRINT " [redacted] Done!"
540 LOCATE 5,5:PRINT "  Well Done!"
550 LOCATE 5,7:PRINT " You have beate
n the"
[redacted] LOCATE 5,9:PRINT "previous hiscor
e."
570 LOCATE 12,13:PRINT "Please type i
n your"
[redacted] LOCATE 12,15:PRINT "name and then
 press"
[redacted] WHILE INKEY$<>"":WEND
600 [redacted] 7,18:PRINT "ENTER"
610 INPUT [redacted]
[redacted] MODE 1
630 hiscore%=score%:score%=0:GOTO [redacted]
640 CLS: LOCATE 12,6:PEN 9:PRINT "You
r score.."score%
650 [redacted] 12,12: PRINT "Today's hisc
ore."hiscore%:LOCATE 12,15:PRINT " b
y ";name$
660 [redacted] delay=1 TO 2500:NEXT delay
670 CLS
[redacted] LOCATE 7,12:PRINT "Do you want to
 play again?":LOCATE 12,15:PRINT "PRE
SS Y OR N "
690 INPUT answer$
```

```
[redacted] INSTR("yY",answer$)<>0 [redacted] CL
S: GOTO [redacted]
710 [redacted]
[redacted] REM ******************************
730 [redacted] title screen set-up **
740 REM ******************************
[redacted] MODE 0
760 [redacted] 0,2
770 [redacted]
780 PEN 10:LOCATE 6,2:PRINT "Welcome
 to  ":LOCATE 5,4:PEN 5:PRINT "LETTER
 LITTER"
[redacted] 1,8: PRINT" A letter [redacted] an
other"
800 LOCATE 1,10:PRINT"symbol will app
ear"
810 LOCATE 1,12:PRINT"on the screen."
820 LOCATE 1,14:PRINT"Press [redacted] right
 key"
830 LOCATE 1,16:PRINT"and a small man
 will"
[redacted] LOCATE 1,18:PRINT"appear. Keep th
e key"
[redacted] LOCATE 1,20:PRINT"pressed [redacted] he
will"
860 [redacted] 1,22:PRINT"collect [redacted] let
ter."
870 LOCATE 4,25:PRINT "Press <SPACE>"
[redacted] IF INKEY(47)<>0 THEN [redacted]
890 CLS
[redacted] LOCATE 1,1: [redacted] 5:PRINT "Don't fo
rget to use"
[redacted] LOCATE 1,3: PEN 5:PRINT "the shif
t key when"
920 LOCATE 1,5:: PEN 5:PRINT "necessa
ry."
930 LOCATE 1,7
[redacted] 1:PRINT "You [redacted] decide the"
950 LOCATE 1,9:PEN 1:PRINT "game leng
th...."
[redacted] 1,13:PEN 5:PRINT "a) [redacted]
conds "
970 LOCATE 1,15:PEN 5:PRINT [redacted] 1 min
ute   "
[redacted] 1,17:PEN 5:PRINT "c) [redacted] min
utes  "
990 [redacted] 1,22:PEN 1:PRINT "press a,
 b or c"
1000 [redacted] INKEY$<>"" [redacted]
[redacted] LOCATE 1,23:PRINT
1020 [redacted] letter$:IF INSTR("aAbBcC",
letter$)=0 GOTO 1010
[redacted] IF letter$="a" [redacted] letter$="A" TH
EN timeallowed =30 ELSE IF letter$="b
" [redacted] letter$="B" [redacted] timeallowed =60
 ELSE [redacted] letter$="c" [redacted] letter$="C" T
[redacted] timeallowed =120
[redacted]
```

Give your fingers a rest . . .
All the listings from this month's issue [redacted] available on cassette.
See our special offer on Page 55.

Turn Grumpies into Smilies in ROLAND WADDILOVE's guessing game

FIVE Grumpies [redacted] standing behind a wall. Can you find out [redacted] colours they [redacted] their order? You can choose how many different colours the Grumpies may [redacted] and you can have up [redacted] 10 [redacted]



Pressing [redacted] of the number keys places a coloured Smiley and DEL will delete the last one. When you have entered a row of five Smileys your guess is marked. A black peg is given for the right colour in the right place, and a white peg for the right colour in the wrong place.

Many of you will have guessed the game it is based on. If you haven't it is very easy to learn and great fun to play, but frustrating when you can't work out the code!

The program is fully structured with no confusing GOTOs. Lines 30 to 50 run the program, calling the various subroutines when necessary. Each of the subroutines has been labelled with a REM statement and they are separated by lines with a single colon.

VARIABLES

code (5)	The secret code.
guess (5)	Your guess.
markedguess (5)	Whether your guess has been marked.
markedanswer (5)	Whether your answer has been marked.
i%, j%	Used in loops.
level	Number of colours.
guesses	Number of guesses made.
correct	Number correct.
k$	Key pressed.
k%	General variable.

SUBROUTINES

Initialise	Defines the tone and volume envelopes and the character for the peg. Dims the arrays, sets the colours and switches off the auto repeat.
Print board	Characters for the board are in data statements. Print the keys and colours used.
Read [illegible] Print	Called to read the data statements and print the board.
Set variables	Set guesses, correct, and level.
Set code	Make [illegible] the secret code.
Input guess	Type in your guess placing Smileys. Delete the last Smiley if DEL is pressed.
Check guess	Mark your guess, see if it is correct.
Award black peg	If the right colour is in the right place.
Award white peg	If the right colour is in the wrong place.
Game over	Print the message and make an appropriate sound. Show the secret code. Ask if you want to play again.
Instructions	Print the instructions.

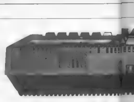
```
[illegible] REM [illegible] Secret Code **
[illegible] [illegible] *By R.A.Waddilove*
[illegible] MODE 1
[illegible] [illegible] 1440
[illegible] MODE 0
[illegible] [illegible] [illegible]
[illegible] WHILE k%=43
[illegible] [illegible] [illegible] [illegible] 710
[illegible] WHILE correct<5 AND guesses<10
[illegible] [illegible] [illegible] [illegible]
110 WEND
120 [illegible] [illegible]
[illegible] [illegible]
[illegible] [illegible] [illegible] 25,2
[illegible] MODE 1:INK 1,24:PEN 1
[illegible] [illegible]
170 :
[illegible] REM ** initialise **
[illegible] [illegible] [illegible] 255,255
[illegible] [illegible] 1,10,1,1
[illegible] ENT 1,100,-1,1
[illegible] ENT 2,100,-1,1,1,0,100,79,1,1
[illegible] [illegible] 3,100,-1,1,1,0,150,95,1,1
[illegible] [illegible] 255,124,124,124,56,56,56,5
6,16
[illegible] DIM code(5),guess(5),markedguess(
[illegible],markedanswer(5)
260 [illegible] i%=0 TO [illegible]
270 READ j%:INK i%,j%
280 NEXT
[illegible] k%=43
[illegible] RETURN
310 :
[illegible] DATA 1,6,9,24,11,15,17,26,7,16,13
,16,3,10,12,0
330 :
[illegible] REM ** print board **
[illegible] [illegible] 8:RESTORE [illegible]
[illegible] [illegible] "    Secret  Code":PRINT
[illegible] [illegible] [illegible]
[illegible] [illegible] i%=10 [illegible] 1 [illegible] -1:PRINT i%:P
[illegible]
390 [illegible] 13:LOCATE 1,2:GOSUB [illegible]
400 [illegible] i%=2 TO 20 STEP 2
410 RESTORE [illegible] 560:GOSUB [illegible]
420 NEXT
[illegible] GOSUB 560:GOSUB 560
[illegible] [illegible] 10:LOCATE 5,23:PRINT "? ? ? ?
 ?"
[illegible] [illegible] 1,25:PEN 14:PRINT "Keys";
[illegible] FOR i%=1 TO level
[illegible] PEN i%:PRINT i%;CHR$(8);
[illegible] NEXT
[illegible] RETURN
[illegible] :
510 [illegible] 150,154,154,158,154,154,154,
154,154,154,154,154,154,158,154,154,1
54,154,154,156
[illegible] [illegible] 149,9,9,149,9,9,9,9,9,9,9,9,
9,149,9,9,9,9,9,149,151,154,154,159,1
54,154,154,154,154,154,154,154,154,15
9,154,154,154,154,154,157
[illegible] [illegible] 149,9,9,149,9,9,9,9,9,9,9,9,
9,149,9,9,9,9,9,149,147,154,154,155,1
54,154,154,154,154,154,154,154,154,15
5,154,154,154,154,154,153
540 :
550 [illegible] [illegible] [illegible] & print **
[illegible] [illegible] k%=1 [illegible] 20
570 [illegible] j%:PRINT CHR$(j%);
580 NEXT
[illegible] [illegible]
600 :
610 [illegible] ** set variables **
620 guesses=0:correct=0:level=0
630 CLS:PEN 11:LOCATE 2,5
[illegible] [illegible] "How [illegible] colours ?"
[illegible] [illegible] 5,10:PRINT "Press 4 - 7"
```

```
[redacted] WHILE level<4 OR level>7
[redacted] level=VAL(INKEY$)
680 WEND
[redacted] RETURN
700 :
710 [redacted] code [redacted]
720 RANDOMIZE TIME
[redacted] i%=1 TO [redacted]
740 code(i%)=INT(RND*level)+1
[redacted]
[redacted] RETURN
770 :
[redacted] REM [redacted] input guess [redacted]
790 WHILE INKEY$<>"":WEND
800 guesses=guesses+1
810 LOCATE 5,23-2*guesses
820 i%=0
[redacted] i%<5
[redacted] i%=i%+1:k$=""
[redacted] WHILE k$<"1" [redacted] k$>CHR$(level+48)
[redacted] k$=INKEY$
870 IF i%>1 [redacted] k$=CHR$(127) THEN PRI
NT CHR$(8);CHR$(8);" ";CHR$(8);:i%=i%
-1
[redacted]
[redacted] SOUND 4,239,5
[redacted] VAL(k$)
[redacted] PRINT CHR$(224);CHR$(9);
920 guess(i%)=VAL(k$)
930 [redacted]
[redacted]
950 :
[redacted] ** check [redacted] **
[redacted] FOR i%=1 TO 5
[redacted] IF guess(i%)=code(i%) [redacted]
1090 ELSE markedanswer(i%)=0:markedgu
ess(i%)=0
[redacted]
[redacted] IF correct=5 [redacted]
[redacted] correct=0
1020 [redacted] i%=1 TO 5
1030 FOR j%=1 TO 5
[redacted] IF guess(j%)=code(i%) AND marked
answer(i%)=0 [redacted] markedguess(j%)=0 TH
EN GOSUB 1160
1050 NEXT
1060 NEXT
[redacted]
1080 :
1090 REM [redacted] award black [redacted]
1100 PEN 15:PRINT CHR$(255);
1110 correct=correct+1
1120 markedanswer(i%)=-1
[redacted] markedguess(i%)=-1
1140 RETURN
```

```
1150 :
1160 [redacted] award white [redacted] **
1170 [redacted] 7;PRINT CHR$(255);
[redacted] markedanswer(i%)=-1
1190 markedguess(j%)=-1
[redacted] RETURN
1210 :
1220 REM ** game over **
1230 PEN 8:PRINT CHR$(30);
1240 IF correct=5 THEN PRINT "   CONG
RATULATIONS":SOUND 4,160,500,3,1,1:SO
UND 2,180,500,2,1,2:SOUND 1,195,500,1
,1,3 ELSE [redacted] 4,1000,500,15:PRINT "
 *** [redacted] LUCK ***"
[redacted] INK 8,7,1
1260 [redacted] i%=1 TO [redacted]
1270 INK 8,7
[redacted] PRINT CHR$(30);"    Secret [redacted]
"
1290 LOCATE 3,23:PEN 12:INK 12,0,1
[redacted] CHR$(243);CHR$(9);:PAPER 1
[redacted]
[redacted] i%=1 TO 5
1320 [redacted] code(i%)
[redacted] PRINT CHR$(225);CHR$(9);CHR$(7);
1340 [redacted] j%=1 TO 2000:NEXT
1350 NEXT
[redacted] LOCATE 1,25:PEN 8:PAPER [redacted]
1370 PRINT "   Another game ? ";:k%=0
[redacted] WHILE [redacted]
[redacted] k%=(43 [redacted] INKEY(43))-1)+(46 AND
INKEY(46))-1)
[redacted] WEND
1410 WHILE INKEY$<>"":WEND
1420 [redacted]
1430 :
[redacted] REM [redacted] instructions **
1450 [redacted] 0,1:INK 1,1:INK 2,18:INK 3,2
6
1460 BORDER 1:PAPER 0:PEN 1:CLS
1470 :LOCATE 1,25
[redacted] PRINT "Secret  Code"
1490 FOR i%=0 TO 192 [redacted] 2
[redacted] FOR j%=2 TO 16 STEP [redacted]
1510 IF TEST(i%,j%) THEN PLOT 200+i%,
366+j%*2,3:PLOT 200+i%,368+j%*2,3
1520 NEXT
1530 NEXT
1540 LOCATE 1,25:PRINT SPACE$(20);
[redacted] INK 1,24:PEN 1:LOCATE 1,5
[redacted] PRINT "Five Brumpies [redacted] hidden
behind [redacted] wall.":PRINT:PRINT "Can you
find out [redacted]at colours they are ?"
1570 PEN 2:PRINT
[redacted] PRINT "After placing five Smiley
's using the":PRINT:PRINT"number keys
 your guess will [redacted] marked.":PRINT:PR
INT"( Delete will remove [redacted] Smiley )."
[redacted] PEN 1:PRINT:PRINT
[redacted] black [redacted] means the rig
ht colour is in":PRINT "the right pla
ce, [redacted] white [redacted] means":PRINT:PRIN
T "t[redacted] right colour is in the wrong [redacted]
lace."
1610 PEN 3:LOCATE 12,25
1620 PRINT "( Press Space )";
[redacted] INKEY$<>"":WEND
[redacted] INKEY$<>" [redacted]
[redacted] RETURN
```

Give your fingers [redacted] rest . . .

All the listings from this month's issue are available on cassette. See our special offer on Page 55.



Amstrad's p
to the big

THE Amstrad CPC464 is capable of supporting all manner of devices which we can attach to it from the outside. But before we can do this we have to know a little bit about what goes [illegible] inside the computer.

At the back of the machine are three connectors to which we can attach devices. These are labelled "User Ports", "Floppy Disc" and "Printer". Although these names indicate what you CAN put in it you are by no means limited to just those devices.

Let's take a look at each connector in turn and [illegible] what potential they have.

The User Port is designed to be connected [illegible] a pair of joysticks and [illegible] scanned along with the keyboard. That means you can read the state of any individual input by using an INKEY command. The codes returned [illegible] printed [illegible] Page 16 of Appendix III of the User Instructions book.

The way this is put together inside the machine means it can only be used as an input, unlike many other computers' user ports which can [illegible] used either [illegible] an input or an output.

Nevertheless many interesting devices may be connected to this port such [illegible] pressure mats for burglar alarms or an analogue to digital converter.

The printer port allows you to connect [illegible] a standard centronics parallel printer interface – the type used by most printers. However you could also [illegible] it as a digital output for many devices.

Normally a printer has an [illegible] bit output but in the case of the Amstrad it only provides a 7 bit output [illegible] the connector marked D7 (pin 9 of the 34-way edge connector) is connected to earth.

While this will not affect normal printing of letters and numbers, some printers use the output to obtain special effects. The Epson FX80 for example uses it to select an alternative italic font.

It is also used in the graphics mode to specify the number of points to print and the state of the eighth needle. This [illegible] be circumvented by programming but it does make it harder than necessary.

In fact, certain printers which cannot be made to alter their line spacings will not be able to produce graphics dumps.

The printer latch [illegible] located at address &FE00 and can be accessed by using the OUT command. This [illegible] be used to switch all manner of devices on and off such as lights and electric kettles.

However [illegible] the voltage coming out of the socket is only [illegible] volts this will have to [illegible] beefed up by extra components before it [illegible] switch anything useful. [illegible] can also be used to control other electronic devices directly like a small robot or [illegible] digital [illegible] analogue converter.

Finally [illegible] the connector marked "Floppy Disc". This is potentially far [illegible] useful that any of the other connectors [illegible] it allows [illegible] to almost the entire bus of the microprocessor.

In addition it allows us access to the light pen input of the 6845 CRT controller chip, which controls the generation of [illegible] video display.

By feeding the signal from a light pen into the chip it will register the position on the screen where the pen is pointing. This can be used for drawing or pointing at menus and is an easy way to interact with your computer.

Inside the chip are certain registers which may be altered to give almost instantaneous sideways scrolling. This is used to great effect in the types of games where you are given a side [illegible] of your rocket and have to guide it over mountains and around tunnels while blasting away – and not being blasted in return.

As we have access to the whole bus of the microprocessor we can also graft extra devices into its memory map. The Z80 microprocessor inside the Amstrad computer is [illegible] of the few microprocessors to have a separate input/output address space.

Normally in [illegible] computer system memory address spaces are used for input/output devices. This means we can have [illegible] maximum amount of RAM and ROM to store our programs and still have all the input and output we want.

It also means that the Z80 provides special instructions to input and output data and does not rely [illegible] the conventional memory access instructions.

The Z80 microprocessor was designed to have 256 bytes of

First [illegible] a series by MIKE COOK on [illegible] horizons with add [illegible]

s ports - back door
g outside world

input/output space but the designers of the Amstrad computer [illegible] to have improved upon that.

Certain input/output instructions place one of the internal registers (the B register) on the higher [illegible] bits of the address bus, which in effect gives us 64k of input/output space. However the designers then seem to have squandered this improvement by making each device take up 256 bytes of address space by [illegible] pletely decoding the address bus.

The result is that we have apparently gained nothing and lost the [illegible] of some important input/output instructions such [illegible] those which can transfer a block of memory directly to an output port.

This strange behaviour may have a rational explanation. It may serve to simplify the circuit, thus reducing the component cost by almost £1. What a price to pay!

When we are considering designing any add-ons in the address space we have to make sure that [illegible] do not conflict with any devices which use this space inside the computer.

The manual which accompanies the computer makes a passing reference to some of the locations used. Table 1 contains a fuller list.

Note that we are not restricted to decoding the higher order address lines so [illegible] have a large amount of space for our devices. But

Address	Use
&F4XX	Port A of the internal D8255
&F5XX	Port B of the internal D8255
&F6XX	Port C of the internal D8255
&F7XX	Control register of the internal D8255
&F8XX to &FBXX	Page expansion bus *
&FEXX	Printer output latch
&DFXX	Expansion ROM
&BCXX	6845 CRT Controller address latch
&BDXX	6845 CRT Controller data latch

** Indicates future expansion. This device is not fitted to the standard computer.*

X indicates that any hex digits may be used and the device will still respond. The digits used have no effect at all so they might as well be 00.

Table 1: The internal input/output address map

remember we cannot [illegible] those powerful input/output instructions lurking in the Z80 as [illegible] would upset the inside of the machine.

In [illegible] designing [illegible] input/output port is simple. We monitor all the address lines until the combination [illegible] have chosen for our device appears. If [illegible] this time the I/O line tells us this [illegible] input/output operation all [illegible] have to do is to place data onto the data bus or record the state of the bus depending upon what the read/write line is telling us to do.

There are many large scale integrated circuits designed to do this. One low cost one is the D8255. This is useful to examine as it is the same type that is used inside the Amstrad to look after the keyboard, user port input and sound chip.

Figure 1 shows the internal configuration of this device. You will see that [illegible] essentially consists of three ports, which means we have three separate 8-bit input/output address locations. The fourth location is a control register which effectively tells the three ports how to behave.

It is [illegible] complex device and can be configured in [illegible] number of different ways. Each one [illegible] called a mode of

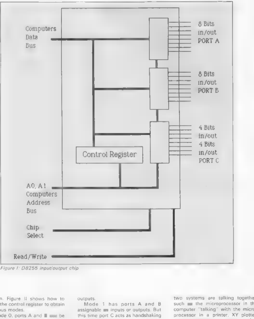


Figure 1: D8255 input/output chip

operation. Figure II shows how to write to the control register to obtain the various modes.

In Mode 0, ports A and [illegible] [illegible] be defined as simple inputs or outputs. Port C is split up into two 4-bit halves, each half of which [illegible] be an input [illegible] an output.

In this simple mode of operation you can read the state of the logic bits on the inputs or change the state of the logic bits by writing to the outputs.

Mode 1 has ports A and B assignable [illegible] inputs or outputs. But this time port C acts as handshaking lines to the two ports.

A handshaking line [illegible] [illegible] line that tells us something about the data [illegible] the ports. It helps transfer the data by indicating when fresh data is available or when a device is not ready to accept data.

Handshaking is used mainly when two systems are talking together, such [illegible] the microprocessor in the computer "talking" with the microprocessor in a printer, XY plotter, storage system or even another computer.

Mode 2 is available only on port A. This makes the port act as a bi-directional data port, which means data can be sent and received from the one port.

In this mode port C provides the

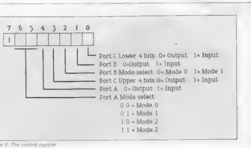


Figure II: The control register

handshaking. This is useful for connecting the data buses of two computers together. Note that when port A is in mode 2 then port B may be in Mode 0 or Mode 1.

Finally there is a method of setting or resetting any individual bit in port C. To do this you write to the control port with a logic one in the most significant bit and a zero in the least significant bit if you want to reset a port C bit, or a one if you want to set it.

The number of the bit you want to alter is placed in bits 1, 2 and 3. Figure III illustrates this operation.

As I said, there is one of these devices lurking inside the Amstrad computer. I think it is safe to assume that it is operating in Mode 0.

As a matter of interest, the BUSY signal from the printer port which tells the computer to hold up any new data while it prints the last data fed into this port. If you want to find it, it goes to bit 6 on port B.

You could use this in your programs to see if the printer is on line before you send it any data.

The Amstrad is a fairly new computer with not many add-ons available for it yet. However this will change.

You may even decide to make some yourself, in which case "watch this space" for complete projects and hints on designing your own.

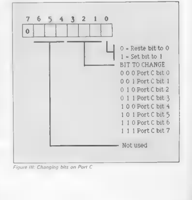


Figure III: Changing bits on Port C

al's beat

ALAN McLACHLAN starts [illegible] regular column prowling the whacky whimsical world of the Amstrad CPC464

I KNEW there would be trouble when she saw it [illegible] the middle of the dining room table.

"What on earth have you bought now?" she asked with [illegible] air of complete bewilderment reminiscent of the [illegible] I turned up with a moped after popping out to buy [illegible] paper.

"It's an Amstrad CPC464", said I with forced nonchalance

"What do you want with another music centre?" she asked. "It's not a music centre love, it's a computer", I replied steadily.

"Not another one?" she slammed, glancing at the console snugly housing my BBC Micro. "What [illegible] you doing – collecting them? Mind you the screen's nicer than the other".

I must admit though, the canary yellow text liberally sprinkled across a bright blue background did brighten the place up [illegible] bit. It added [illegible] beautiful backdrop [illegible] the condiment set and the cracked vase from granny that [illegible] daren't throw away for fear of reprisals from her gang of senile delinquents.

I don't collect computers, but they have become both my work and play. I am no expert programmer but I have limited experience of two other machines.

I bought a 1k ZX80 three years ago when I was a copper. Well, someone had to. Seriously, it was cheap enough to have [illegible] play with and I could throw it away if I didn't take to it. But of course, I did – hook, line and sinker.

Within [illegible] fortnight though I was fed up with running out of memory – *(and the micro was [illegible] bit short of RAM [illegible] well! – Ed).* We needed to expand. I sold it to the newspaper lad and bought, in my opinion, the most economical keyboard computer around at the time – the Dragon 32k! Just think of all that extra space.

I got a lot of pleasure out of the Dragon and became reasonably proficient at Basic. Since then I've progressed onto the BBC Micro and finally the Amstrad. Which is where we [illegible] in.

When I arrived home with it I was bubbling with excitement. A 64k

'Why oh why is the inclusion of a [illegible] plug such a daunting prospect [illegible] manufacturers?'

micro, [illegible] cassette recorder and a colour monitor all in [illegible] package was [illegible] offer I just could not refuse.

I unpacked the hardware with trembling fingers. At last, with protective polystyrene scattered to the four corners of the room and the cat making [illegible] bed out of the discarded plastic bags, it was there in [illegible] its glory – plugless!

Why, oh why, is the inclusion of a 5 amp plug such [illegible] daunting prospect to the manufacturers of many electrical components? Where do you get [illegible] from [illegible] 6.30pm [illegible] a Friday?

Of course I should have anticipated this omission (I do at Christmas with battery-operated thingies for the kids), but I made light of it, literally. There is now no plug on the sideboard lamp.

I'm not behind the door when it [illegible] to matters electrical. I did, however, make a slight – what they would call in the trade – technical faux pas. Before attaching the wire to the terminals, I forgot to pass them through the hole in the top of the plug. I've got more plugs at home with that hole cut through than I would care to mention. I don't think I'm alone either.

I connected the keyboard to the monitor, plugged in and switched on. And switched on again. The screen was a delight and the character set looked nice too. I curbed my natural desire to start typing and sat for [illegible] few minutes with the manual, just in case I had missed anything on my first glance. Considering that this had been in the pub a couple of days previously I thought better safe than sorry!

The first useful point that struck

me was the apparent ability to type in lower case – variables and keywords alike. I tried it.

```
10 cls
[illegible] print"hello"
run
hello
Ready
```

You see, I've read lots of beginners articles and according to cocker you should always print "hello" at your first attempt [illegible] the keyboard. So far so good! Right we'll list it.

```
l.
Syntax error
```

First mistake. The BBC accepted abbreviated commands (L. was short for LIST) but the Amstrad doesn't. I wasn't bothered about it but I must have made the same mistake [illegible] least [illegible] times during the first week. Old habits die hard. Try again, Al.

```
list
[illegible] CLS
20 PRINT"hello"
```

It worked. As predicted the command words, because they [illegible] "tokenised" – they are recognised by the computer and allocated [illegible] particular code – appeared as capitals.

This proved to be a very useful aspect of the machine because identifying incorrectly spelt commands became simple. And being fairly simple I typed a lot of incorrect commands! They remained in lower case, silent witnesses to my incompetence.

```
10 cls
[illegible] print"hello"
[illegible] printt"hello"
[illegible]
hello
Syntax error in [illegible]
[illegible] printt"hello"
```

What's happened [illegible] that [illegible] micro has executed the program up to the error and [illegible] only indicated the error line but has also listed it in edit mode. By this I mean that the line [illegible] [illegible] ready for editing using the cursor keys. I'll explain this later when I understand it myself.

The important thing [illegible] notice here is that because "printt" was spelt wrongly, it did not reappear [illegible] PRINT in the listed line.

This is [illegible] extremely useful facility, particularly when entering long listings. Normally my programs have more bugs in them than [illegible] down and outs hostel and to be able to spot syntax errors quickly is a great de-bugging aid.

And while we're [illegible] the word PRINT I'll mention another interesting point. I'd typed it in numerous times before I discovered that there [illegible] an abbreviated form. I'd been trying [illegible] fathom out the colour commands you know:

```
The ink is black,
The paper's white,
Wrong way round and
It's out of sight.....
```

Well I've sussed it out now (I think!) and I'll tell you about the experiments another day. At the time though, I gave [illegible] up as a bad job and took the manual to bed. (There's [illegible] way a manual can [illegible] a headache as an excuse for non-cooperation.)

I started at the beginning again and found the helpful hint about the ? replacing the PRINT command. I was tempted to leap out of bed and dash back to the machine, but managed to curb my enthusiasm when I glanced at the clock – 2am.

The ? is very handy and particularly useful in replacing those infernal P. syntax errors. And of course, it comes out [illegible] PRINT when the program is listed. (How does it know (PRINT))

```
10 cls
20 ?"Hello again"
list
10 CLS
[illegible] PRINT"Hello again"
```

This abbreviation is not new to me as I came [illegible] it [illegible] the Dragon.

Another useful abbreviation [illegible] the Dragon which is also used on the Amstrad [illegible] the REM command. The computer ignores everything after a REM and this is extremely useful for making little notes within your programs to make them [illegible] to follow.

```
10 cls
[illegible] PRINT"Bonjour":REM French for
  Hello
30 PRINT"Hello":  ' short for [illegible]
```

The abbreviation is on the shifted 7 key ('). It does save [illegible] few letters of typing, but is not quite [illegible] obvious as the full statement. Because of this, listings in *Computing with the Amstrad* will use the full REM rather than the abbreviation and you would be well advised to follow suit.

Well, I think we'd better call it a draw for this month. All this new-fangled technology is a bit heavy [illegible] the eyelids and I really should get my beauty sleep. *(and boy, does he need it -Ed.)*

'I typed [illegible] [illegible] lot of incorrect commands . . . they remained in lower case, silent witnesses to incompetence'

BOOKSHOP

A choice selection of 24 [redacted] roaring games that'll give you hours [redacted] fun.
It also shows you how to incorporate ready-made routines into your own programs.

£6.95

[redacted] the most of your Amstrad's potential [redacted] this information-packed book.
It's [redacted] easy to [redacted] information that's certain to improve your programming.

£6.95

Starting where the Users Instructions leave off [redacted] invaluable book explores the Amstrad's sound, graphics and assembly language capabilities to the full.

£8.95

Put your Amstrad to good use with these forty games, [redacted] of educational value. Subjects include languages, geography, [redacted] and science.

£6.95

If you're baffled by the Users Instructions, this is the book for you.
Its easy-to-follow no-nonsense introduction to Amstrad Basic [redacted] highly [redacted]ended.

£7.95

[redacted] you've [redacted] wondered what [redacted] behind the [redacted] in adventure games, or thought about writing one, this [redacted] thought-out book is the [redacted] for you.

£7.95

From adventures to asteroid storms, dungeons to dice, the 31 programs featured in this book [redacted] guaranteed to give you plenty of amusement - plus valuable insights [redacted] programming techniques.

£6.95

ORDER FORM

All prices include post and packing

Please enter number required in box

			£
☐	40 Educational Games for the Amstrad by Vince Apps (Granada)	£6.95	....
☐	[redacted] Amstrad Program [redacted] by Peter [redacted] (Phoenix)	£6.95	....
☐	Adventure Games for [redacted] CPC464 by A.J. Bradbury (Collins)	£7.95	....
☐	Amstrad Computing by [redacted] (Granada)	£6.95	....
☐	Sensational Games for the [redacted] by [redacted] Gregory (Granada)	£6.95	....
☐	The Amstrad [redacted] Explored (Kuma)	£8.95	....
☐	The Amstrad CPC464 Advanced User [redacted] by Mark Harrison (Signal Press)	£7.95	....

I enclose my cheque/P.O for £ ....

Name ....

Address ....

....

Post to: Database Publications, Europa House, 68 Chester Road Hazel Grove, Stockport SK7 5NY

Cheques made payable to Database Publications Ltd.

These prices apply to [redacted] readers only. Overseas prices on request

WATCH THIS SPACE FOR NEW BOOKS EVERY MONTH, [redacted] DEVELOP YOUR KNOWLEDGE OF THIS [redacted] MACHINE STILL FURTHER!

Give your fingers a rest!

ALL Computing with the AMSTRAD **programs on one cassette for only £3.75**

Why tire your fingers typing when you can get all the [redacted] from this issue in [redacted] value-packed cassette?

Take a look at what's on offer and you'll see what we [redacted] We've games galore, useful utilities – plus lots of exciting extras . . .

SMILEY: *Can you avoid the Grumpies [redacted] you guide Smiley round the labyrinth? Our Amstrad version of this arcade favourite guarantees hours of fun.*

CODE: *You don't have to be [redacted] mastermind [redacted] play this intriguing logic game – but it helps!*

BINARY: *Baffled by binary bits? Let our utility help you out.*

DANCER: *Simple but fun, our dancer is a lovely little mover.*

TRAPPER: *You'll need quick-thinking, fast reactions and downright cunning if you're going to successfully pen the malevolent monster of the maze.*

SCROLLER: *Add that professional touch to your text with this slick sideways scrolling routine.*

LETTER LITTER: *Keep your Amstrad tidy and learn the keyboard layout at the same time with this entertaining educational game.*

PLUS all 13 example programs from our Sound and Graphics articles.

You've read the mag now load the tapes!

HOW TO ORDER

Please send me the cassette of programs from the January issue of *Computing with the Amstrad*

☐ I enclose cheque for £3.75 made payable to Database Publications Ltd

☐ I wish to pay by: ☐ Access ☐ Visa

No. ______

Expiry date ______

Signed ______

Name ______

Address ______

POST TO: Amstrad Tape Offer, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.

It's all done by numbers

MIKE BIBBY helps make sense of machine code

I THINK the question I get asked most often by micro enthusiasts is: *"What exactly is machine code? I just can't make sense of all this LD (HL),nn and JP NZ business. I bought [redacted] book, but that didn't help".*

Well this series of articles is [redacted] attempt to answer that question. You may not be an accomplished machine code programmer at the end of it, but you will certainly know what machine code is, and be able to write your own simple programs.

Better than that, you'll be in [redacted] position to take advantage of the many excellent books on Z80 machine code currently on the market, and see how they fit in with your Amstrad. From then on you'll [redacted] able to teach yourself, and that's always the best way.

So what IS [redacted] machine code program?

Well, let me dodge the question by telling you that all programs are machine code, eventually, and we'll get round to exactly what that means in a minute.

First of all, tradition decrees that I tell you that the microprocessor at the heart of the Amstrad CPC464 is the Z80A, complete with [redacted] and 16 bit registers and [redacted] 16 bit address bus. I should then go on to discuss its arithmetic logical unit, its internal data bus and [redacted] on, referring you to [redacted] incomprehensible diagram showing its "architecture".

To heck with all that. Let's talk about it from the consumer's point of view – yours. You [redacted], I'm not one of those "you can drive a car better if you know what's under the bonnet" freaks. I have it on good authority that gynaecologists do not make the best lovers.

So what is machine code all about? The fact is, it's all about numbers – lots of them. More precisely, it's about lots of numbers, each of which is between 0 and 255 in value.

Show me [redacted] machine code program and I'll show you a load of such numbers. Forget about LD and JP for the moment. Believe me, it's all done by numbers.

Let me explain. We're used to talking about a micro having memory aren't we. Well a micro's memory is composed of lots of individual memory cells, as is our own brain. And, just like our memory cells, a micro's memory cell can only remember so much.

In the [redacted] of the Z80, the cell [redacted] remember only [redacted] byte [redacted] a time – and a byte, you won't be surprised to learn, happens to [redacted] number in the range 0 [redacted] 255.

An upper limit of 255 might seem [redacted] little arbitrary, but there's an excellent reason for it. It's all to do with the wiring (Okay, we'll lift the bonnet just [redacted] little!)

Each memory cell, or location [redacted] it's [redacted] properly termed, consists of [redacted] of eight switches, each of which can be either ON or OFF. Now by arranging the switches [redacted] various patterns of on and off, [redacted] encode things – a sort of electrical semaphore. And what [redacted] code [redacted] – yes, you've guessed it – numbers!

Have a look at Table I. What it does [redacted] link each switch with a number. We've labelled [redacted] switches switch 0, then switch 1 and so on up to switch 7. Notice that, yet again, computers start counting at [redacted]. Even though we only go up to switch 7, there [redacted] eight switches in all.

Now below each switch [redacted] the table is the number linked to it (don't worry why we picked these particular numbers for the moment). Switch 0 is worth 1, switch 1 is worth 2, switch 2 is worth 4 and [redacted] on.

Notice how the value of each switch doubles [redacted] you go along. Given these values we can code numbers. For example, if switch 4 were ON, and all the others off, we'd be "hiding" the number 16. Similarly, if just switch 7 were [redacted], we'd be coding 128.

Even better, by having more than one switch on at a time [redacted] can code other numbers than just the eight we've linked so far – [redacted] just add the values of all the switches that are ON, to arrive at the [redacted] number. For instance, if switch 5 and switch 1 were on simultaneously, and all the others were off, the number we've got is 34. Figure I shows why.

If you think about it for [redacted] moment, you'll [redacted] that the smallest number [redacted] encode is 0 (all the switches OFF) and the largest number we can encode is 255 (all the switches on).

The really nice thing about the way we've chosen our numbers though, is that every number between 0 and 255 has its own unique pattern of switches, so there's never any confusion about the number you've coded, or stored – to use computerese – in the byte.

But, and it's a big but, writing 34 as

Switch	7	6	5	4	3	2	1	0
Value	128	64	32	16	8	4	2	1

Table I: Values associated with each switch

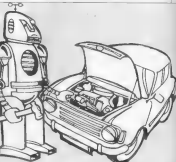
Switch	7	6	5	4	3	2	1	0
Value	128	64	32	16	8	4	2	1
State	off	off	on	off	off	off	on	off
gives - - - - > 32 + 2 = 34								

Figure I: Encoding 34 with switches

off off on off off off on off is incredibly cumbersome. However mathematicians decided that since there were only two states for each switch (ON and OFF), they'd use the number 1 for ON and 0 for OFF. Using 1 and 0 in this way gives us what are called binary numbers. In this scheme of things 106 becomes %01101010. Figure II shows how.

You may be wondering why we've put the % in front of the 01101010. The reason is that otherwise we might mistake it for an incredibly large ordinary number. So if you see [redacted] % in front of a number it's coded in our binary way. Incidentally, each switch [redacted] known [redacted] a bit, and since [redacted] byte consists of eight such switches, we can say that there [redacted] eight bits in [redacted] byte. The article Bits and Bytes on Page 38 goes into it in more detail.

Now these eight bits in a memory byte allow us [redacted] store any number from 0 to 255 - 256 different numbers, [redacted] you remember to count 0. But if we're going to have a computer of any power we're going to need more than 256 bytes of memory.



So what the micro does is to have 65536 different memory locations, numbered from 0 to 65535, to store its data in. Why 65536? Well, in order [redacted] keep track of its memory bytes, the computer has to do some more wiring.

We've already [redacted] that having eight wires would only allow us to keep tabs on 256 locations. What the Z80 does is to double up the number of wires to 16 – which then gives it 65535 as its largest number. Look at Table II if you don't believe me.

As you can see, it's like the old Table I with [redacted] extra eight switches or bits added on top - that is, another byte.

To get the value of these extra bits [redacted] just keep on doubling. 128 was the last [redacted], so it goes 256, 512 and so [redacted] up to 32768. The top (higher valued) set of eight bits is called the high byte of the address - hi byte for short. The bottom (lower valued) set of eight [redacted] called the low byte of the address - lo byte for short.

If [redacted] the switches [redacted] on – that is, if all the bits [redacted] set [redacted] 1 – the number these two bytes would code is:

```
 %1111111111111111
= 32768 +
  [illegible]
   8192
   4096
   2048   hi byte values
   1024
    512
  [illegible]
    128
     64
     32
     16   lo byte values
      8
      4
      2
      1
 -------
  65535
```

Now do you believe me? The reason we've gone into so much detail is because, as I've said, machine code is all about numbers stored in the micro's memory. In fact machine code is mostly about moving those numbers (and hence the information encoded in them) around the memory of the computer – that is,

hi byte	bit	15	14	13	12	11	10	[redacted]	[redacted]
	value	32768	16384	8192	4096	2048	1024	512	256
lo byte	bit	7	[redacted]	5	4	3	2	1	0
	value	128	64	32	16	8	4	2	1

Table II: 16 bits explained – compare with Table I

Switch	7	6	5	4	3	2	1	0
Value	128	64	32	16	[redacted]	4	2	1
State	off	on	on	off	on	off	on	off
binary	0	1	1	0	1	0	1	0

Giving - - - -> 64 + 32 + 8 + 2 = 106 decimal
- - - -> %01101010 binary

Figure II: The binary representation of 106

from one memory location to another.

For example, there's a large machine code program that actually runs your CPC464. It's called the operating system, or firmware. One of its jobs is to print that familiar "welcome" message on the screen when you first turn on the machine.

What happens is that the message [illegible] stored away in the micro's memory. When you switch on it copies the message from those locations into the memory reserved for the screen [illegible] you can see it. That is, the firmware machine code program moves the numbers that encode the message from one location to another.

This same sort of transfer of data occurs when you press a key. The firmware transfers the value of the key pressed from the location that remembers which key it was to the memory set aside for the screen.

When you save [illegible] Basic program the firmware's own machine code program moves the contents of the memory where the Basic program is stored out to the cassette port.

It's all about moving bytes of data around! More formally, most of machine code is concerned with moving bytes of information from [illegible] memory location to another. If you're a realist, you'll probably have guessed that there's a lot more to it than that. But have faith, most of what I'm telling you is true.

To investigate this movement of bytes further we need [illegible] analogy – in other words a meaningful lie. Suppose we have a tiny micro with only three memory locations. Figure III shows the sort of thing.

It's fairly easy to wire them up so that the numbers can move from one location to another – just join each byte to every other byte (in the figure,

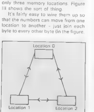


Figure III: Linking three memory locations

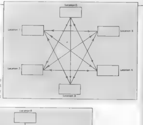


Figure IV: The complexities of linking six memory locations

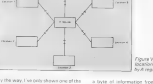


Figure V: Memory locations linked by A register

by the way, I've only shown one of the eight wires for clarity).

[illegible] you stretch your imagination you'll [illegible] that it looks a lot like a simple railway.

But suppose there were more locations, as in Figure IV. You can see the layout's getting complicated. And when you consider that the Z80 addresses tens of thousands of such locations, you can see that we've got problems – the wiring's far too complex.

Of course the answer is to stop giving each memory location its own direct lines to each and every other location. We'll do what railways do, and have junctions and branch lines.

Figure V shows such [illegible] layout. Our six locations are all connected via the major junction A. Everything passes through here. In fact there is such a memory location [illegible] A – [illegible] major junction through which traffic passes. It's deep in the heart of the Z80 and can hold one byte numbers. In computing jargon we call such [illegible] junction [illegible] register.

Now suppose you wanted to move a byte of information from memory location [illegible] to memory location [illegible]. As you can see from the figure, you would go via register – that is, junction – A.

You would do this by giving the machine two instructions:

1. *Load register A with the number contained in memory location zero.*

2. *Load memory location 5 with the number that is in register A.*

It's [illegible] sort of microelectronic pass the parcel. The data goes from location 0 to A, then from A to 5. It's always a two-stage journey. All traffic passes through A.

In practice the actual layout is more like Figure VI, but there's still no direct traffic. Everything goes to A first and then back out.

To stretch our analogy a little further, a major rail junction like A would have lots of facilities that other junctions haven't. It's the same with the Z80 – once you've got a number in A you can do all sorts of things with it!

Also no rail designer worth his salt would depend on one major junction

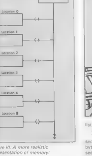
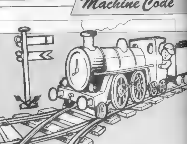


Figure VI: A more realistic representation of memory/register linkage



– there'd be too much congestion. He'd have other junctions. Similarly the Z80 has registers other than A for much the same [redacted].

And [redacted] it stands [redacted] junction/register A doesn't have all that much capacity – just [redacted] eight bit number. This could be limiting if [redacted] wanted to deal with larger numbers, such as we use to label memory locations. Well the Z80 has got registers to handle these, too, [redacted] we'll see later on in the series.

By now I think I've convinced you that machine code's all about moving numbers around in memory. But how does the micro know what to do? How do you tell it to move these numbers, and where you want them putting?

The answer's simple – you give it [redacted] list of numbers stored in memory!

I'm not joking, honest.

The program itself is just a sequence of bytes in memory. The bytes have meaning to the Z80, you see – it's a sort of code: machine code in fact. All you do [redacted] point your [redacted] the first byte and say go. It then [redacted] along the list of bytes doing what it's told.

Let's have a look at what this [redacted] in the context of the little program we discussed above – the [redacted] that transferred [redacted] byte from location 0 to location 5.

The actual string of bytes we need [redacted]:

58 0 0 50 5 0 [redacted]

I've written the numbers in decimal, [redacted] that's what we're used to – of course the micro reads them in binary. Figure VII explains what it's all about. When we point the Z80 at the location of the first byte of our program and tell it to go it knows that first byte is going to tell it to do something. The fancy name for this sort of "command" byte is [redacted] opcode – short for operation code.

Now 58 is an opcode that tells the Z80 to load register A with the contents of a particular location in memory. From the opcode itself, the Z80 knows that the address of this memory location will be contained in the two bytes directly following the opcode (remember you need two bytes to specify addresses).

So having understood the meaning of the opcode, the Z80 moves its attention along to these two bytes and works out the address they refer to (in this case, location 0). It then copies the contents of that address into the A register.

The Z80 has finished with the first three bytes and has done all the first opcode instructed it to. It now turns its attention to the fourth byte, which it knows must be an opcode since it has finished its previous task.

This time the byte is 50, which tells the micro to load the memory location specified in the next two bytes with the number in the A register. (In a sense, this is the mirror image of the last opcode. That loaded the accumulator from a memory

Contents of memory location	58	00	00	50	05	00	201
Meanings of above bytes	Load A with contents of the address that follows	These two bytes specify the address needed by previous opcode		Load the address following with the contents of A register	These two bytes specify the address referred to by the previous opcode		Return from whence you came

Figure VII: A simple machine code program explained

location. This opcode loads a memory location from A.)

So having worked out what the opcode contained in the fourth byte wants it to do, the Z80 turns its attention to the next two bytes along, works out the address they store and copies the A register into that location.

Having finished that instruction, which used the fourth, fifth and sixth bytes, the Z80 then moves on to the seventh and last byte to find its next opcode.

The seventh byte contains 201, the opcode for return – which tells the Z80 to go back to where it was before it started or, as the jargon has it, called this progam.

This works in much the [redacted] way [redacted] RETURN does in [redacted] subroutine, causing the micro to rejoin the main flow of the program.

Notice that you don't need any extra bytes after this opcode to tell it where [redacted] return to. When this routine [redacted] called the Z80 carefully stored where it was up to for future reference – as does [redacted] Basic program when it meets [redacted] GOSUB.

By the way, you may have noticed that the two bytes specifying location five are not 0.5 as you might expect, but 5.0.

I don't want to go [redacted] this too much this month. Suffice it to say that the Z80 likes to know the lo byte of an address before it receives the hi byte.

Let's have another look at the machine code program we've developed. I'm going to split each instruction – that [redacted] each opcode and its data bytes – onto a separate line.

```
58 0 0
50 5 0
201
```

Doesn't make immediate sense, does it? Our brain is much [redacted] adept at making sense of words than numbers. Have a look [redacted] the program [redacted] new form, that [redacted] "words":

```
LD A,(0)  58 0 0
LD (5),A  50 5 0
RET       201
```

The symbols [redacted] the lefthand side are mnemonics. LD stands for LoaD and RET for RETurn.

The translation is [redacted] follows:

LD A,(0) *LoaD the A register with the contents of memory location 0.*

LD (5),A *LoaD memory location 5 with the contents of register A.*

RET *RETurn to the program that called the machine code in the first place.*

You can get special programs called assemblers that let you type in your routines in these more meaningful mnemonics and then translate them into machine code, but they're a luxury we'll be doing without for a while.

Well that's all for now. I hope you've got [redacted] better understanding of what machine code is.

● *Next month we'll be looking at hexadecimal and running your own machine code programs. Until then, take a look [redacted] Bits [redacted]d Bytes [redacted] Page 38. And practice your binary – you'll be needing it!*

Going for a scroll

SCROLLER is a short, simple program that scrolls a message letter by letter from right [illegible] left [illegible] the screen, repeating it endlessly in a "wrap around" display.

When you run the program it asks you for the message you w[illegible] displayed and the line you'd like it to scroll across. The program does the rest, displaying the message until you press a key.

If you want to incorporate Scroller in your own programs it couldn't be easier. The subroutine that does the work (GOSUB 270) can just be renumbered and merged with your programs. All you have to do is to set up the two variables mentioned in its REM statements.

Pete Bibby

SUBROUTINES

GOSUB 90 Checks and accepts the message to be displayed, adding [illegible] space to it. [illegible] also accepts the line the scrolling is to take place on, again trapping erroneous input.

GOSUB 270 Does the actual work using two FOR ... NEXT loops. The first loop has the message appearing from the right, growing letter by letter, travelling towards the left. This loop stops when the whole message has completed its journey from right to left and is displayed [illegible] the screen.

The next loop carries [illegible] the scroll, displaying the message in the centre of the line, giving a wrap around effect. This [illegible] achieved by taking a letter off the front of the message, adding it to the end of what's left and then overprinting the original with this [illegible] string. The delay loops of lines 400 and 480 can be left out if you want to see how fast your micro [illegible] work.

VARIABLES

message$, submessage$ Hold the string to be scrolled plus a space which separates the messages.

aline, subeline Determine the screen line that the display is to use.

sublength The length of the message.

subdisplay$ At first this is full of spaces, but each time the first loop cycles a space is removed and a letter added in its place until it eventually holds the whole message. The second loop uses it to contain and display the dismembered message [illegible] it wraps around.

substartposition Holds the position of the left end of the display, which is fixed in line 340 so that it's centred [illegible] the line.

```
10 REM SCROLLER
20 REM PETE BIBBY
[illegible] REM (c) [illegible] PUBLICATIONS
[illegible] MODE 1
50 [illegible] 90:REM Accept message
[illegible] GOSUB 270:REM Scroll message
70 END
80 REM**********************
90 REM Accept and check message
100 LOCATE 1,10
110 PRINT "What message do you want s
crolled?"
[illegible] LOCATE 1,12
[illegible] INPUT message$
140 IF LEN(message$)>38 [illegible] [illegible]
150 REM put a space at the end of the
 message
160 message$=message$+" "
170 LOCATE 1,20
180 PRINT "Which line do you want it
to appear on?"
190 LOCATE 1,22
[illegible] INPUT aline
210 IF aline<1 OR aline>25 GOTO 170
[illegible] submessage$=message$
230 subline=aline
[illegible] CLS
250 RETURN
[illegible] REM********************
270 REM Scroll the message
[illegible] [illegible] the subroutine needs these [illegible]
riables [illegible] called:
[illegible] REM subline holds the line positi
on
[illegible] REM submessage$ holds the string
[illegible] REM [illegible] delay loops slow things [illegible]
[illegible]
320 sublength=LEN(submessage$)
[illegible] subdisplay$=STRING$(sublength," "
)
340 substartposition=INT((40-sublengt
h)/2+1)
350 REM first loop has string appeari
[illegible] from right
[illegible] FOR loop=1 TO sublength
370 subdisplay$=LEFT$(subdisplay$,(su
blength-loop))+LEFT$(submessage$,loop)
[illegible] LOCATE substartposition, subline
390 PRINT subdisplay$
[illegible] [illegible] delay=1 TO 500:NEXT delay
410 NEXT loop
[illegible] REM second loop just cycles the o
riginal message
430 [illegible] until [illegible] key is pressed
440 WHILE INKEY$=""
[illegible] subdisplay$=RIGHT$(subdisplay$,(s
ublength-1))+LEFT$(subdisplay$,1)
[illegible] LOCATE substartposition, subline
[illegible] PRINT subdisplay$
[illegible] FOR delay=1 TO 500:NEXT delay
490 WEND
500 RETURN
510 REM********************
```

Give your fingers [illegible] rest ...

All the listings from this month's issue are available on cassette. See our special offer on Page 55.

Starting at ground zero

AMSTRAD Computing with the CPC464 is the latest offering from the prolific pen of Ian Sinclair. Aimed at the newcomer to the Amstrad, the beginner to programming of any sort is certainly not forgotten.

Mr Sinclair says in his opening paragraph: "The Amstrad CPC464 computer offers much more for either the home or the business user than any previous machine at [redacted] comparable price, more [redacted] than many machines at much higher prices. Because of this many buyers of the machine will never have made use of a computer before, certainly not [redacted] computer with the range of facilities that the CPC464 offers".

He certainly [redacted] impressed with it, but is not prepared to ignore trivialities such as finding the key tops loose when he opened the hardware package for the first time. He made it sound the most natural thing in the world and included an explanation of how to put the spacebar back on with the aid of [redacted] pair of tweezers.

The book starts off like all good quality beginner's books should – at "ground zero". The first chapter covers first and foremost, in detail, how to attach a five amp plug. Why not? The manufacturers couldn't be bothered to fit one, so let's get things right from the start! Connecting to a monitor/TV, and tuning the signal is next on the agenda.

This is followed by [redacted] short introduction to some of the command keys such [redacted] Esc, Del, Enter and Ctrl and a comprehensive discussion of loading and saving.

The book then takes you gently through the PRINT command using numbers and strings and an explanation of the functions TAB and LOCATE. In no time at all you are using string and number variables with INPUT, and even defining [redacted] simple function in the following manner:

```
5 [redacted] sum(A,B,C)=A+B+C
10 CLS
[redacted] PRINT"Enter three numbers,please"
[redacted] INPUT A,B,C
[redacted] PRINT "Sum is ";FNsum(A,B,C)
```

Nothing earth-shattering here I must admit, but the way it [redacted] explained makes it natural [redacted] find such "advanced" techniques so early in the book.

There's [redacted] excellent chapter [redacted] loops which starts off with the dreaded GOTO and proceeds through FOR . . . NEXT to the completely [redacted] – WHILE . . . WEND.

As Mr Sinclair states: "Very few home computers offer you any [redacted] than [redacted] simple FOR . . . NEXT loop . . . the CPC464 however offers you a very different kind of loop. This [redacted] often make it much easier to program".

I have yet to find this out. I trust him implicitly and will persevere. Mind you, it [redacted] much easier to fall back on [redacted] FOR . . . NEXT when things get tricky.

There are sizeable chapters on menus, subroutines and file handling, and all are explained in an interesting, easy-to-read [redacted]. I have still fully to get to grips with file handling but this book certainly goes a long way to making it a little clearer for me.

The graphics section covers three chapters and gives a comprehensive account of the WINDOW, BORDER, PAPER, PEN and INK commands.

In chapter 10, "Guide to greater graphics", the MOVE, PLOT and DRAW commands [redacted] explained with excellent examples. However nowhere could I find the answer to the one question that has been bugging me since I started programming the CPC464: "How do you set [redacted] foreground colour while PRINTing at the graphics cursor (TAG), without using the PLOT command with its colour parameter?"

I can't find any other solution than plotting [redacted] point off screen. I'm [redacted] I'll be enlightened soon by someone who has found it oh [redacted] simple.

The CPC464 has vast SOUND potential with its two envelopes for both sound and tone, and Mr Sinclair gives quite [redacted] successful detailed explanation of both.

It [redacted] as easy to understand as such [redacted] computer subject can ever be if it can be, and I feel more confident in tackling these facilities having read this exposition.

To cap it all the book rounds off with appendices [redacted] printing, programming keys to do special things and error trapping.

The many useful and well annotated listings were all taken straight from the Amstrad and therefore should contain no errors. Having heard this claim before I was somewhat sceptical, but must admit that all the examples I tried worked perfectly.

All in all this book would make [redacted] useful supplement to the Amstrad's own manual and at £6.95 is good value for money.

Alan Sergeant

Amstrad Computing with the CPC464 *(Granada, £6.95).*

Let's keep track of those sub-routines

I THOUGHT you would be interested in a little hint for keeping track of subroutines without searching through the whole program.

The idea is to have all the sub-routines' first line numbers listed at the start of the program. The list consists of GOSUB statements to point to the routines. For example the program may look like this:

```
10 GOTO 100
20 GOSUB 2000: REM
   Create characters.
30 GOSUB 3000: REM
   Initialise screen.
40 GOSUB 4000: REM
   Instructions.
50 GOSUB 5000: REM
   Move invaders.
60 GOSUB 6000: REM
   Move base.
70 GOSUB 7000: REM
   Fire laser.
80 GOSUB 8000: REM
   Highscore table.
90 GOSUB 9000: REM
   Game over.
100 REM ****START OF
   PROGRAM****
2000 REM Characters.
3000 REM Screen.
```

When the program is renumbered the list of GOSUBs is updated. The GOTO at the start of the program jumps round this list and therefore the GOSUBs are never executed.

I hope this will be of some use to your readers. – **Terry Blunt, Berkshire.**

■ This is a very simple tip but very, very useful. We will certainly be making use of it while developing our own programs – we are sure many of our readers will too.

Amstrad disc drives ... on their way soon

Discs and modems

I HAVEN'T got an Amstrad yet, but I am seriously considering buying one as it seems to be a very competitively priced package.

I was wondering whether you could clear up a couple of points that would help me make up my mind.

I don't want to buy it if there is any risk that the disc system may never appear. Have you any comments on this?

Secondly, in this, the year of communications, I hope soon to purchase a modem. Will the Amstrad support one?

These questions may seem trivial, but I don't want to go and lash out over £200 and then find out that it wasn't what I had hoped. – **David Ingles, Shrewsbury.**

■ We have it from a reliable source that both disc drives and communications hardware will be launched shortly.

Future of Forth

I HAVE spent nearly 12 months getting the hang of Forth on a Jupiter Ace. Alas it is no more, and has gone to that great chip shop in the sky.

I understand the Amstrad will support high level languages, and I am thinking of buying one. Is there likely to be a Forth for it, and if so will it be in ROM or on cassette? – **Julie Bayliss, Wexford.**

■ We have spoken to Amsoft and they say Forth is not in their immediate plans, but it could well be on the cards for the near future.

It is a reasonably safe bet that if Amsoft don't do it someone else will.

Buffer bother

I LEARNT to program on a BBC Micro at school but now I've got myself an Amstrad and I'm recovering.

The problem I've got is how to empty the keyboard buffer. On the Beeb this is easy but I can't find any way of doing it on the CPC.

As it is, I keep getting stuff INPUTed at the keyboard that I don't want. Any ideas? – **Ian Worth, Glasgow.**

■ To be honest, no. There must be a simple way of doing it but all we can think of is to use a line such as:

```
WHILE INKEY$<>"" : WEND
```

just before you go to the keyboard to look for an input. This clears out any garbage and leaves the micro ready to receive fresh input.

Anybody got any better ways?

Sound advice

ONE of your writers has informed me that you're doing a magazine for the Amstrad CPC464. If so, this could be your first letter.

Can you answer a question for me? I've been messing around with the SOUND command but things keep going out of step.

The noises I make (or, rather, the micro makes) are out of synch with the program. I've only been using channel A.

Any ideas on how to

Computing with the AMSTRAD Postbag

WE welcome letters from readers – about your experiences using the CPC464, about tips you would like to pass on to other users ... and about what you would like to see in future issues.

The address to write to is:

Postbag Editor
Computing with the Amstrad
Europa House
68 Chester Road
Hazel Grove
Stockport SK7 5NY

remedy it? – **Tom Price, Wolverhampton.**

■ Surprisingly you're not our first letter, our fame seems to have gone before us! As for your problem, it sounds as though what's happened is that you've discovered the sound queues.

As you know, Amstrad Basic is very fast and it can whip through a short program in a fraction of a second. The trouble is that we usually want the sounds we make to last for a second or two, but we don't want the program to stop while they're playing.

The CPC gets round this by popping the notes on a sound queue which processes them independently while it gets on with the Basic.

Your problem occurs when you want the program to coincide with a sound at a certain point in its execution.

You can do this using the SQ function which tests the state of a channel. A line like:

```
WHILE SQ(1)>= 127 :WEND
```

holds up the program by cycling round until channel A is free. You can use this as a means of making the end of a note coincide with the next bit of Basic.

Epson link-up

I HAVE connected an Epson FX80 to my Amstrad CPC464 with the help of a knowledgeable person, and have encountered a few problems.

Each time I list my programs I get double line feeds, by this I mean a blank line between each program line.

Another problem is that character codes sent to the printer greater than 127 are mis-interpreted.

Have you any ideas as to why the Amstrad is rejecting the printer? – **Paul Fisher, Bristol.**

■ The problem with line feeds can be solved by cutting wire 14 of your printer ribbon cable. Actually finding the fourteenth can be a little tricky. Starting at the side with the marked wire, the wires go 1, 19, 2, 20, 3, 21 and so on. If you've counted correctly the wire 14 should be the eighth in from the unmarked side.

Epson FX80: why should the Amstrad reject it?

As you correctly state, all code greater than 127 produce unusual effects. This is because bit 7 of each byte sent to the printer is ignored. This has the same effect as sending the character ANDed with 127.

What makes it worse is that both the hardware and software allow only 7 bits.

If any reader knows how to get round this problem, please write in with the solution. By the way, there's a screen dump for the FX80 coming up in a future issue of *Computing with the Amstrad*.

Random reflections

AS the owner of a BBC I decided recently to invest in the Amstrad as a second computer.

I was experimenting with the RND(X) function and noticed that whatever number I enclosed in the brackets the micro responded with a six figure decimal number.

If this is correct why does the manual quote RND(6) as an example when RND(anything) would achieve the same result?

It would appear that the only way to get random whole numbers is to use the INT function on a random number that has been multiplied by 10 or 100 depending on the range required. – **Matthew Gould, Winchester.**

■ The RND function only generates a number between 0 and 0.999999999, and this is the case whatever you include in the brackets.

The ideal way to solve your problem is to use the expression X = INT(number*(RND(1)) + 1 where "number" is the highest number in the range you require.

This will generate random numbers between 1 and "number". If you need the range to start at 0 the expression must be changed to X = INT((number+1) * RND(1)).

Colour quandary

CAN you help me? I'm absolutely desparate.

When I'm developing my own programs I often change the values of the pen/ink combinations.

When the inevitable errors occur, and I break into my programs to list them, the listing often appears in quite hideous colours.

What I want is a way to get back to the colour set that's there at switch on.

I know I can do this with Ctrl + Shift + Esc, but I lose my program.

Do you know how to do it? – **Sandra Brown, Solihull.**

■ What you need to do is CALL &BC02 and all will be well. You can set up the small Enter key to do this for you with:

```
KEY 139, "CALL &BC02" +
CHR$(13)
```

We tend to link it up with pen as follows:

```
KEY 139, "CALL &BC02 : PEN
1 " + CHR$(13)
```

We also put a call to &BC02 at the end of our programs – it's only common politeness to leave things tidy!

ACCESSING THE 6845 CHIP

I'VE had a BBC for about one year and have just bought a CPC464. I have heard from a reliable source that the Amstrad has got a 6845 Cathode Ray Tube Controller (CRTC) – also found in the BBC computer.

I know how to program this chip directly on a BBC but I cannot find any way of doing it on my Amstrad. Do you know how this can be done? – **Frank Henderson, Manchester.**

■ From Basic you can access the 6845 chip by using the OUT command.

The 6845 Command Register is programmed by OUT &BC00,X, where X is the register you wish to program.

The Parameter Register is programmed in a similar way except that the OUT &BD00,Y command is used – Y is the number to be written to the previously selected register.

The 6845 is a very powerful chip. Games which use sideways scrolling effects rely on the 6845 for speedy movement.

We hope to cover the subject of the 6845 in a future issue of *Computing with the Amstrad*.

Below is a simple program which shows how the 6845 can be programmed to change the start location of the video RAM – this is done by writing to registers 12 and 13 of the 6845.

```
10 OUT &BC00,12
20 OUT &BD00,63
30 OUT &BC00,13
40 OUT &BD00,0
```